Chapter 11

## Requirements

**Learning Objectives**

After studying this chapter, you should be able to

* + - Perform the requirements workflow.
    - Draw up the initial business model.
    - Draw up the requirements.
    - Construct a rapid prototype.

The chances of a product being developed on time and within budget are somewhat slim unless the members of the software development team agree on what the software product is to do. The first step in achieving this unanimity is to analyze the client’s current situation as precisely as possible. For example, it is inadequate to say, “The client needs a computer-aided design system because they claim their manual design system is lousy.” Unless the develop- ment team knows exactly what is wrong with the current manual system, there is a high probability that aspects of the new computerized system will be equally “lousy.” Similarly, if a personal computer manufacturer is contemplating development of a new operating system, the first step is to evaluate the firm’s current operating system and analyze carefully exactly why it is unsatisfactory. To take an extreme example, it is vital to know whether the problem exists only in the mind of the sales manager who blames the operating system for poor sales, or whether users of the operating system are thoroughly disenchanted with its functionality and reliability. Only after a clear picture of the present situation has been gained can the team attempt to answer the critical question, What must the new product be able to do? The process of answering this question is the primary objective of the requirements workflow.

**11.1 Determining What the Client Needs**

Một quan niệm sai lầm thường được cho là, trong quy trình làm việc yêu cầu, các nhà phát triển phải xác định phần mềm mà khách hàng muốn. Ngược lại, mục tiêu thực sự của quy trình làm việc yêu cầu là xác định phần mềm nào khách hàng cần. Một vấn đề là

**313**



**Just in Case You Wanted to Know**

**Box 11.1**

S. I. Hayakawa (1906–1992), U.S. Senator from California, once told a group of reporters, “I know you believe you understood what you think I said, but I am not sure you realize that what you heard is not what I meant.” This excuse applies equally well to the issue of requirements analysis. The software engineers hear their client’s requests, but what they hear is not what the client should be saying.

That quotation has been wrongly attributed to former U.S. presidential candidate George Romney (1907–1995) who once announced at a press conference, “I didn’t say that I didn’t say it. I said that I didn’t say I said it. I want to make that very clear.” Romney’s “clarification” highlights another challenge of requirements analysis—it is easy to misunderstand what the client says.

Nhiều khách hàng không biết họ cần gì. Hơn nữa, ngay cả một khách hàng có ý tưởng tốt về những gì cần thiết cũng có thể gặp khó khăn trong việc truyền đạt chính xác những ý tưởng này cho các nhà phát triển vì hầu hết khách hàng ít hơn máy tính biết chữ hơn các thành viên của nhóm phát triển.(For more insight into this issue, see Just in Case You Wanted to Know Box 11.1.)

Một vấn đề khác là khách hàng có thể không đánh giá cao những gì đang diễn ra trong tổ chức của chính họ. Ví dụ, không có ích gì khi khách hàng yêu cầu một sản phẩm phần mềm nhanh hơn khi lý do thực sự tại sao sản phẩm phần mềm hiện tại có thời gian phản hồi lâu như vậy là cơ sở dữ liệu được thiết kế kém. Những gì cần phải làm là tổ chức lại và cải thiện cách dữ liệu được lưu trữ trong sản phẩm phần mềm hiện tại, nếu không một sản phẩm phần mềm mới sẽ chỉ là chậm như vậy. Hoặc, nếu khách hàng điều hành một chuỗi cửa hàng bán lẻ không có lợi nhuận, khách hàng có thể yêu cầu một hệ thống thông tin quản lý tài chính phản ánh các khoản mục như bán hàng, tiền lương , các khoản phải trả và các khoản phải thu. Một hệ thống thông tin như vậy sẽ ít được sử dụng nếu lý do thực sự cho những tổn thất là co rút (trộm cắp và trộm cắp của nhân viên). Nếu đúng như vậy, thì cần phải có một hệ thống kiểm soát cổ phiếu thay vì một hệ thống thông tin quản lý tài chính.

Ngay từ cái nhìn đầu tiên, việc xác định những gì khách hàng cần rất đơn giản — các thành viên của nhóm phát triển chỉ cần hỏi họ. Tuy nhiên, có hai lý do tại sao cách tiếp cận trực tiếp này thường không hoạt động tốt.

Đầu tiên, như vừa được nêu, khách hàng có thể không đánh giá cao những gì đang diễn ra trong tổ chức của mình. Nhưng lý do chính khiến khách hàng thường yêu cầu sản phẩm phần mềm sai là phần mềm phức tạp. Thật khó để một kỹ sư phần mềm hình dung ra một sản phẩm phần mềm và chức năng của nó — vấn đề còn lâu mới xảy ra đối với khách hàng, người thường không phải là chuyên gia về kỹ thuật phần mềm.

Nếu không có sự hỗ trợ của một nhóm phát triển phần mềm lành nghề, khách hàng có thể là một nguồn thông tin kém về những gì cần được phát triển. Mặt khác, trừ khi có giao tiếp trực tiếp với khách hàng, không có cách nào để tìm ra những gì thực sự cần thiết.

Nỗ lực cổ điển trong việc giải quyết thách thức này được mô tả trong Phần 11.12. Cách tiếp cận hướng đối tượng là lấy thông tin ban đầu từ khách hàng và người dùng trong tương lai của sản phẩm mục tiêu và sử dụng thông tin ban đầu này làm đầu vào cho quy trình làm việc yêu cầu of the Unified Process [Jacobson, Booch, and Rumbaugh, 1999]. This is described in Section 11.2.

* 1. Overview of the Requirements Workflow

The overall aim of the **requirements workflow** is for the development organization to determine the client’s needs. The first step toward this goal is to gain an understanding of the **application domain** (or **domain**, for short), that is, the specific environment

in which the target product is to operate. The domain could be banking, space explora- tion, automobile manufacturing, or telemetry. Once the members of the development team understand the domain to a sufficient depth, they can build a business model, that is, use UML diagrams to describe the client’s business processes. The business model is used to determine what the client’s initial requirements are. Then iteration is applied.

In other words, the starting point is an initial understanding of the domain. This information is used to build the initial business model. The initial business model is utilized to draw up an initial set of the client’s requirements. Then, in the light of what has been learned about the client’s requirements, a deeper understanding of the domain is gained; and this knowledge is utilized in turn to refine the business model and hence the client’s requirements. This iteration continues until the team is satisfied with the set of requirements. At this point, the iteration stops.

The term **requirements engineering** is sometimes used to describe what is performed during the requirements workflow. The process of discovering the client’s requirements is termed **requirements elicitation** (or **requirements capture**). Once the initial set of requirements has been drawn up, the process of refining and extending them is termed **requirements analysis**.

We now examine each of these steps in detail.

(Dịch: Mục đích chung của quy trình làm việc yêu cầu là để tổ chức phát triển xác định nhu cầu của khách hàng. Bước đầu tiên hướng tới mục tiêu này là hiểu biết về miền ứng dụng (gọi tắt là miền), tức là môi trường cụ thể mà sản phẩm mục tiêu sẽ hoạt động. Lĩnh vực này có thể là ngân hàng, thám hiểm không gian, sản xuất ô tô hoặc đo từ xa. Sau khi các thành viên của nhóm phát triển hiểu miền đủ sâu, họ có thể xây dựng mô hình kinh doanh, nghĩa là sử dụng sơ đồ UML để mô tả các quy trình kinh doanh của khách hàng. Mô hình kinh doanh được sử dụng để xác định yêu cầu ban đầu của khách hàng là gì. Sau đó, lặp lại được áp dụng.

Nói cách khác, điểm khởi đầu là sự hiểu biết ban đầu về miền. Thông tin này được sử dụng để xây dựng mô hình kinh doanh ban đầu. Mô hình kinh doanh ban đầu được sử dụng để thiết lập các yêu cầu ban đầu của khách hàng. Sau đó, dựa trên những gì đã học được về các yêu cầu của khách hàng, bạn sẽ hiểu sâu hơn về miền; và kiến ​​thức này lần lượt được sử dụng để tinh chỉnh mô hình kinh doanh và do đó là các yêu cầu của khách hàng. Việc lặp lại này tiếp tục cho đến khi nhóm hài lòng với các yêu cầu đặt ra. Tại thời điểm này, quá trình lặp lại dừng lại.

Thuật ngữ kỹ thuật yêu cầu đôi khi được sử dụng để mô tả những gì được thực hiện trong quy trình làm việc theo yêu cầu. Quá trình khám phá các yêu cầu của khách hàng được gọi là kích thích các yêu cầu (hoặc nắm bắt các yêu cầu). Khi tập hợp các yêu cầu ban đầu đã được thiết lập, quá trình tinh chỉnh và mở rộng chúng được gọi là phân tích yêu cầu.

Bây giờ chúng ta kiểm tra chi tiết từng bước này.)

* 1. Understanding the Domain

To elicit the client’s needs, the members of the requirements team must be familiar with the application domain, that is, the general area in which the target product is to be used. For example, it is not easy to ask meaningful questions of a banker or a neurosurgeon without first acquiring some familiarity with banking or neurosurgery. Therefore, an initial task of each member of the requirements analysis team is to acquire familiarity with the applica- tion domain, unless he or she already has experience in that general area. It is particularly important to use correct terminology when communicating with the client and potential users of the target software. After all, it is hard to be taken seriously by a person working in a spe- cific domain unless the interviewer uses the nomenclature appropriate for that domain. More important, use of an inappropriate word may lead to a misunderstanding, eventually resulting in a faulty product being delivered. The same problem can arise if the members of the require- ments team do not understand the subtleties of the terminology of the domain. For example, to a layperson words like *brace, beam, girder*, and *strut* may appear to be synonyms, but to a civil engineer they are distinct terms. If a developer does not appreciate that a civil engineer is using these four terms in a precise way and if the civil engineer assumes that the developer is familiar with the distinctions among the terms, the developer may treat the four terms as equivalent; the resulting computer-aided bridge design software may contain faults that result in a bridge collapsing. Computer professionals hope that the output of every program will be scrutinized carefully by a human before decisions are made based on that program, but the growing popular faith in computers means that it is distinctly unwise to rely on the likelihood of such a check being made. So, it is by no means far-fetched that a misunderstanding in ter- minology could lead to the software developers being sued for negligence.

One way to address the problem with terminology is to construct a **glossary**, a list of technical words used in the domain, together with their meanings. The initial entries are inserted into the glossary while the team members are busy learning as much as they can

about the application domain. Then, the glossary is updated whenever the members of the requirements team encounter new terminology. Every so often, the glossary can be printed out and distributed to team members or downloaded to a PDA (such as a Palm Pilot or Black- Berry). Not only does such a glossary reduce confusion between client and developers, it also is useful in lessening misunderstandings between the members of the development team.

Once the requirements team has acquired familiarity with the domain, the next step is to build the business model.

* 1. The Business Model (Mô hình kinh doanh)

A **business model** is a description of the business processes of an organization. For example, some of the business processes of a bank include accepting deposits from clients, loaning money to clients, and making investments.

The reason for building a business model first is that the business model provides an understanding of the client’s business as a whole. With this knowledge, the developers can advise the client as to which portions of the client’s business to computerize. Alternatively, if the task is to extend an existing software product, the developers have to understand the existing business as a whole to determine how to incorporate the extension and to learn what parts, if any, of the existing product need to be modified to add the new piece.

To build a business model, a developer needs to obtain a detailed understanding of the various business processes. These processes are now *refined*, that is, analyzed in greater detail. A number of different techniques can be used to obtain the information needed to build the business model, primarily interviewing

(Dịch: Mô hình kinh doanh là một mô tả về các quy trình kinh doanh của một tổ chức. Ví dụ, một số quy trình kinh doanh của ngân hàng bao gồm nhận tiền gửi từ khách hàng, cho khách hàng vay tiền và đầu tư.

Lý do để xây dựng mô hình kinh doanh trước tiên là vì mô hình kinh doanh cung cấp sự hiểu biết về toàn bộ hoạt động kinh doanh của khách hàng. Với kiến ​​thức này, các nhà phát triển có thể tư vấn cho khách hàng về những phần nào trong công việc kinh doanh của khách hàng cần tin học hóa. Ngoài ra, nếu nhiệm vụ là mở rộng một sản phẩm phần mềm hiện có, các nhà phát triển phải hiểu toàn bộ hoạt động kinh doanh hiện tại để xác định cách kết hợp tiện ích mở rộng và tìm hiểu những phần nào của sản phẩm hiện có cần được sửa đổi để thêm vào mảnh mới.

Để xây dựng một mô hình kinh doanh, một nhà phát triển cần có hiểu biết chi tiết về các quy trình kinh doanh khác nhau. Các quy trình này hiện đã được tinh chỉnh (refined), tức là được phân tích chi tiết hơn. Một số kỹ thuật khác nhau có thể được sử dụng để có được thông tin cần thiết để xây dựng mô hình kinh doanh, chủ yếu là phỏng vấn.)

* + 1. Interviewing

The members of the requirements team meet with members of the client organization until they are convinced that they have elicited all relevant information from the client and future users of the target software product.

There are two basic types of questions. A closed-ended question requires a specific answer. For example, the client might be asked how many salespeople the company employs or how fast a response time is required. Open-ended questions are asked to encourage the person being interviewed to speak out. For instance, asking the client, “Why is your current software product unsatisfactory?” may explain many aspects of the client’s approach to business. Some of these facts might not come to light if the question were closed ended.

Similarly, there are two basic types of interviews, structured and unstructured. In a **structured interview**, specific preplanned questions are asked, frequently closed ended. In an **unstructured interview**, the interviewer may start with one or two prepared closed- ended questions, but subsequent questions are posed in response to the answers he or she receives from the person being interviewed. Many of these subsequent questions are likely to be open ended in nature to provide the interviewer with wide-ranging information.

At the same time, it is not a good idea if the interview is too unstructured. Saying to the client, “Tell me about your business” is unlikely to yield much relevant knowledge. In other words, questions should be posed in such a way as to encourage the person being interviewed to give wide-ranging answers but always within the context of the specific information needed by the interviewer.

Conducting a good interview is not always easy. First, the interviewer must be fully familiar with the application domain. Second, there is no point in interviewing a member

of the client organization if the interviewer has already made up his or her mind regarding the client’s needs. No matter what the interviewer has previously been told or what he or she has learned by other means, the interviewer must approach every interview with the intention of listening carefully to what the person being interviewed has to say, while firmly suppressing any preconceived notions regarding the client company or the needs of the client and the potential users of the target product to be developed.

After the interview is concluded, the interviewer must prepare a written report outlining the results of the interview. It is strongly advisable to give a copy of the report to the person who was interviewed; he or she may want to clarify certain statements or add overlooked items.

(Dịch: Các thành viên của nhóm yêu cầu gặp gỡ các thành viên của tổ chức khách hàng cho đến khi họ tin chắc rằng họ đã lấy được tất cả thông tin liên quan từ khách hàng và những người dùng tương lai của sản phẩm phần mềm mục tiêu.

Có hai loại câu hỏi cơ bản. Một câu hỏi đóng yêu cầu một câu trả lời cụ thể. Ví dụ: khách hàng có thể được hỏi công ty sử dụng bao nhiêu nhân viên bán hàng hoặc yêu cầu thời gian phản hồi nhanh như thế nào. Các câu hỏi mở được đưa ra để khuyến khích người được phỏng vấn nói ra. Ví dụ, hỏi khách hàng, "Tại sao sản phẩm phần mềm hiện tại của bạn không đạt yêu cầu?" có thể giải thích nhiều khía cạnh trong cách tiếp cận kinh doanh của khách hàng. Một số sự thật này có thể không được đưa ra ánh sáng nếu câu hỏi được kết thúc.

Tương tự, có hai loại phỏng vấn cơ bản, có cấu trúc và không có cấu trúc. Trong một cuộc phỏng vấn có cấu trúc, các câu hỏi cụ thể được lập kế hoạch trước được hỏi, thường kết thúc. Trong một cuộc phỏng vấn không có cấu trúc, người phỏng vấn có thể bắt đầu với một hoặc hai câu hỏi kết thúc đã chuẩn bị sẵn, nhưng các câu hỏi tiếp theo được đặt ra để đáp lại câu trả lời mà họ nhận được từ người được phỏng vấn. Nhiều câu hỏi trong số những câu hỏi tiếp theo này có thể được kết thúc mở để cung cấp cho người phỏng vấn thông tin trên phạm vi rộng.

Đồng thời, sẽ không phải là một ý kiến ​​hay nếu cuộc phỏng vấn quá không có cấu trúc. Nói với khách hàng, "Hãy cho tôi biết về doanh nghiệp của bạn" không có khả năng mang lại nhiều kiến ​​thức liên quan. Nói cách khác, các câu hỏi nên được đặt ra theo cách để khuyến khích người được phỏng vấn đưa ra những câu trả lời trên phạm vi rộng nhưng luôn nằm trong ngữ cảnh của thông tin cụ thể mà người phỏng vấn cần.

Thực hiện một cuộc phỏng vấn tốt không phải lúc nào cũng dễ dàng. Đầu tiên, người phỏng vấn phải hoàn toàn quen thuộc với miền ứng dụng. Thứ hai, không có ích gì khi phỏng vấn một thành viên của tổ chức khách hàng nếu người phỏng vấn đã quyết định về nhu cầu của khách hàng. Bất kể người phỏng vấn đã được nói gì trước đây hoặc những gì họ đã học được bằng các phương tiện khác, người phỏng vấn phải tiếp cận mọi cuộc phỏng vấn với mục đích lắng nghe cẩn thận những gì người được phỏng vấn nói, đồng thời kiên quyết loại bỏ mọi định kiến ​​về công ty khách hàng hoặc nhu cầu của khách hàng và người dùng tiềm năng của sản phẩm mục tiêu được phát triển.

Sau khi cuộc phỏng vấn kết thúc, người phỏng vấn phải chuẩn bị một báo cáo bằng văn bản nêu rõ kết quả của cuộc phỏng vấn. Rất nên đưa một bản sao của báo cáo cho người được phỏng vấn; người đó có thể muốn làm rõ các tuyên bố nhất định hoặc thêm các mục bị bỏ qua.)

* + 1. Other Techniques

Interviewing is the primary technique for obtaining information for the business model. This section describes some other techniques that may be used in conjunction with interviewing.

One way of gaining knowledge about the activities of the client organization is to send a **questionnaire** to the relevant members of the client organization. This technique is useful when the opinions of, say, hundreds of individuals need to be determined. Furthermore, a carefully thought-out written answer from an employee of the client organization may be more accurate than an immediate verbal response to a question posed by an interviewer. However, an unstructured interview conducted by a methodical interviewer who listens carefully and poses questions that elicit amplifications of initial responses usually yields far better information than a thoughtfully worded questionnaire. Because questionnaires are preplanned, there is no way that a question can be posed in response to an answer.

A different way of eliciting requirements is to examine the various **forms** used by the

business. For example, a form in a printing works might reflect press number, paper roll size, humidity, ink temperature, paper tension, and so on. The various fields in this form shed light on the flow of print jobs and the relative importance of the steps in the printing process. Other documents, such as operating procedures and job descriptions, also can be powerful tools for finding out exactly what is done and how. If a software product is being used, the user manuals should also be carefully studied. A comprehensive set of different types of data regarding how the client currently does business can be extraordinarily help- ful in determining the client’s needs. Therefore, a good software professional carefully studies client documentation, treating it as a valuable potential source of information that can lead to an accurate assessment of the client’s needs.

Another way of obtaining such information is by **direct observation** of the users, that is, by members of the requirements team observing and writing down the actions of the employees while they perform their duties. A modern version of this technique is to set up **videotape cameras** within the workplace to record (with the prior written permis- sion of those being observed) exactly what is being done. One difficulty of this technique is that it can take a long time to analyze the tapes. In general, one or more members of the requirements team has to spend an hour playing back the tape for every hour that the cameras record. This time is in addition to what is needed to assess what was observed. More seriously, this technique has been known to backfire badly because employees may view the cameras as an unwarranted invasion of privacy. It is important that members of the requirements team have the full cooperation of all employees; it can be extremely difficult to obtain the necessary information if people feel threatened or harassed. The possible risks should be considered carefully before introducing cameras or, for that matter, taking any other action that has the potential to annoy or even anger employees.

(Dịch: Phỏng vấn là kỹ thuật chính để thu thập thông tin cho mô hình kinh doanh. Phần này mô tả một số kỹ thuật khác có thể được sử dụng cùng với phỏng vấn.

Một cách để đạt được kiến ​​thức về các hoạt động của tổ chức khách hàng là gửi bảng câu hỏi cho các thành viên có liên quan của tổ chức khách hàng. Kỹ thuật này hữu ích khi ý kiến ​​của hàng trăm cá nhân cần được xác định. Hơn nữa, một câu trả lời bằng văn bản được suy nghĩ cẩn thận từ một nhân viên của tổ chức khách hàng có thể chính xác hơn một câu trả lời bằng lời nói ngay lập tức cho câu hỏi do người phỏng vấn đặt ra. Tuy nhiên, một cuộc phỏng vấn không có cấu trúc được thực hiện bởi một người phỏng vấn có phương pháp, người lắng nghe cẩn thận và đặt ra những câu hỏi khơi gợi sự khuếch đại của những câu trả lời ban đầu thường mang lại thông tin tốt hơn nhiều so với một bảng câu hỏi được viết kỹ lưỡng. Bởi vì bảng câu hỏi được lập kế hoạch trước, không có cách nào mà một câu hỏi có thể được đặt ra để đáp ứng với một câu trả lời.

Một cách khác để đưa ra các yêu cầu là kiểm tra các hình thức khác nhau mà doanh nghiệp sử dụng. Ví dụ: một biểu mẫu trong tác phẩm in ấn có thể phản ánh số lần ép, kích thước cuộn giấy, độ ẩm, nhiệt độ mực, độ căng của giấy, v.v. Các trường khác nhau trong biểu mẫu này làm sáng tỏ luồng công việc in và tầm quan trọng tương đối của các bước trong quy trình in. Các tài liệu khác, chẳng hạn như quy trình vận hành và mô tả công việc, cũng có thể là công cụ mạnh mẽ để tìm ra chính xác những gì được thực hiện và cách thức thực hiện. Nếu một sản phẩm phần mềm đang được sử dụng, hướng dẫn sử dụng cũng cần được nghiên cứu cẩn thận. Một tập hợp toàn diện các loại dữ liệu khác nhau về cách khách hàng hiện đang kinh doanh có thể giúp ích rất nhiều trong việc xác định nhu cầu của khách hàng. Do đó, một chuyên gia phần mềm giỏi sẽ nghiên cứu kỹ lưỡng tài liệu của khách hàng, coi nó như một nguồn thông tin tiềm năng có giá trị có thể dẫn đến đánh giá chính xác về nhu cầu của khách hàng.

Một cách khác để thu thập thông tin đó là quan sát trực tiếp người dùng, nghĩa là các thành viên của nhóm yêu cầu quan sát và viết ra các hành động của nhân viên trong khi họ thực hiện nhiệm vụ của mình. Một phiên bản hiện đại của kỹ thuật này là thiết lập các máy quay băng video trong nơi làm việc để ghi lại (với sự cố định bằng văn bản trước của những người được quan sát) chính xác những gì đang được thực hiện. Một khó khăn của kỹ thuật này là có thể mất nhiều thời gian để phân tích các cuộn băng. Nói chung, một hoặc nhiều thành viên của nhóm yêu cầu phải dành một giờ để phát lại đoạn băng mỗi giờ mà camera ghi lại. Thời gian này ngoài những gì cần thiết để đánh giá những gì đã quan sát được. Nghiêm trọng hơn, kỹ thuật này được cho là có tác dụng ngược vì nhân viên có thể coi camera như một hành vi xâm phạm quyền riêng tư không chính đáng. Điều quan trọng là các thành viên của nhóm yêu cầu có sự hợp tác đầy đủ của tất cả nhân viên; có thể rất khó để có được thông tin cần thiết nếu mọi người cảm thấy bị đe dọa hoặc quấy rối. Các rủi ro có thể xảy ra nên được xem xét cẩn thận trước khi giới thiệu máy ảnh hoặc, đối với vấn đề đó, thực hiện bất kỳ hành động nào khác có khả năng làm phiền hoặc thậm chí tức giận nhân viên.)

**FIGURE 11.1**

**Banking Software Product**

Withdraw Money

The Withdraw Money use case of the banking software product.

**Customer**

**Teller**

* + 1. Use Cases

As stated in Section 3.2, a **model** is a set of UML diagrams that represent one or more aspects of the software product to be developed (recall that the *ML* in UML stands for “modeling language”). A primary UML diagram used in business modeling is the use case.

A **use case** models an interaction between the software product itself and the users of that software product (**actors**). For example, Figure 11.1 depicts a use case from a banking software product. There are two actors, represented by the UML stick figures, the **Customer** and the **Teller**. The label inside the oval describes the business activity rep- resented by the use case, in this instance Withdraw Money.

Another way of looking at a use case is that it shows the interaction between the soft- ware product and the environment in which the software product operates. That is, an actor is a member of the world outside the software product, whereas the rectangle in the use case represents the software product itself.

It is usually easy to identify an actor.

* An actor is frequently a user of the software product. In the case of a banking software product, the users of that software product are the customers of the bank and the staff of the bank, including tellers and managers.
* In general, an actor plays a role with regard to the software product. This role may be as a user of the software product. However, an initiator of a use case or someone who plays a critical part in a use case is also playing a role and is therefore regarded as an actor, irrespective of whether that person is also a user of the software product. An example of this is given in Section 11.7.

A user of the system can play more than one role. For example, a customer of the bank can be a **Borrower** (when he or she takes out a loan) or a **Lender** (when he or she deposits money in the bank—a bank makes much of its profit by investing the money deposited by customers). Conversely, one actor can participate in multiple use cases. For example, a **Borrower** may be an actor in the Borrow Money use case, the Pay Interest on Loan use case, and the Repay Loan Principal use case. Also, the actor **Borrower** may stand for many thousands of bank customers.

An actor need not be a human. Recall that an actor is a user of a software product, and in many cases another software product can be a user. For example, an e-commerce informa- tion system that allows purchasers to pay with credit cards has to interact with the credit card company information system. That is, the credit card company information system is an actor from the viewpoint of the e-commerce company information system. Similarly, the e-commerce information system is an actor from the viewpoint of the credit card company information system.

**FIGURE 11.2**

Generalization of medical staff.

**Medical Staff**



**Physician Nurse**

As previously stated, identification of actors is easy. Generally, the only difficulty that arises in this part of the paradigm is that an overzealous software professional sometimes identifies overlapping actors. For example, in a hospital software product, having a use case with actor **Nurse** and a different use case with actor **Medical Staff** is not a good idea, because all nurses are medical staff, but some medical staff (such as physicians) are not nurses. It would be better to have actors **Physician** and **Nurse**. Alternatively, actor **Medical Staff** can be defined with two specializations, **Physician** and **Nurse**. This is depicted in Figure 11.2. In Section 7.7, it was pointed out that inheritance is a special case of generalization. Generalization was applied to classes in Section 7.7. Figure 11.2 shows how generalization can be applied to actors, too.

(Dịch: Như đã nêu trong Phần 3.2, mô hình là một tập hợp các biểu đồ UML đại diện cho một hoặc nhiều khía cạnh của sản phẩm phần mềm sẽ được phát triển (nhớ lại rằng ML trong UML là viết tắt của “ngôn ngữ mô hình hóa”). Một sơ đồ UML chính được sử dụng trong mô hình kinh doanh là trường hợp sử dụng.

Một ca sử dụng mô hình hóa sự tương tác giữa bản thân sản phẩm phần mềm và những người sử dụng sản phẩm phần mềm đó (các tác nhân). Ví dụ, Hình 11.1 mô tả một ca sử dụng từ một sản phẩm phần mềm ngân hàng. Có hai tác nhân, được đại diện bởi các số liệu thanh UML, Khách hàng và Người bán. Nhãn bên trong hình bầu dục mô tả hoạt động kinh doanh bị cản trở bởi trường hợp sử dụng, trong trường hợp này là Rút tiền.

Một cách khác để xem xét một ca sử dụng là nó cho thấy sự tương tác giữa sản phẩm phần mềm và môi trường mà sản phẩm phần mềm hoạt động. Nghĩa là, một tác nhân là một thành viên của thế giới bên ngoài sản phẩm phần mềm, trong khi hình chữ nhật trong ca sử dụng đại diện cho chính sản phẩm phần mềm.

Nó thường dễ dàng để xác định một tác nhân.

• Một tác nhân thường là người dùng sản phẩm phần mềm. Đối với sản phẩm phần mềm ngân hàng thì đối tượng sử dụng sản phẩm phần mềm đó là khách hàng của ngân hàng và nhân viên của ngân hàng, bao gồm cả giao dịch viên và cán bộ quản lý.

• Nói chung, một tác nhân đóng một vai trò nào đó đối với sản phẩm phần mềm. Vai trò này có thể là người sử dụng sản phẩm phần mềm. Tuy nhiên, người khởi xướng một ca sử dụng hoặc một người nào đó đóng một vai trò quan trọng trong một ca sử dụng cũng đóng một vai trò nào đó và do đó được coi là một tác nhân, bất kể người đó có phải là người dùng sản phẩm phần mềm hay không. Một ví dụ về điều này được đưa ra trong Phần 11.7.

Một người dùng của hệ thống có thể đóng nhiều vai trò. Ví dụ: khách hàng của ngân hàng có thể là Người đi vay (khi anh ta hoặc cô ta đi vay) hoặc Người cho vay (khi anh ta hoặc cô ta gửi tiền vào ngân hàng — một ngân hàng tạo ra phần lớn lợi nhuận bằng cách đầu tư số tiền mà khách hàng gửi ). Ngược lại, một tác nhân có thể tham gia vào nhiều trường hợp sử dụng. Ví dụ: Bên vay có thể là một tác nhân trong trường hợp sử dụng Tiền đi vay, trường hợp sử dụng Khoản vay phải trả lãi và trường hợp sử dụng Khoản vay trả nợ gốc. Ngoài ra, diễn viên Người đi vay có thể đứng ra đại diện cho hàng nghìn khách hàng của ngân hàng.

Một diễn viên không cần phải là một con người. Nhớ lại rằng một tác nhân là một người sử dụng một sản phẩm phần mềm, và trong nhiều trường hợp, một sản phẩm phần mềm khác có thể là một người dùng. Ví dụ, một hệ thống thông tin thương mại điện tử cho phép người mua thanh toán bằng thẻ tín dụng phải tương tác với hệ thống thông tin của công ty thẻ tín dụng. Có nghĩa là, hệ thống thông tin công ty thẻ tín dụng là một tác nhân theo quan điểm của hệ thống thông tin công ty thương mại điện tử. Tương tự, hệ thống thông tin thương mại điện tử là một tác nhân theo quan điểm của hệ thống thông tin công ty thẻ tín dụng.

Như đã nói trước đây, việc xác định các tác nhân rất dễ dàng. Nói chung, khó khăn duy nhất nảy sinh trong phần này của mô hình là một chuyên gia phần mềm quá chăm chỉ đôi khi xác định các tác nhân chồng chéo. Ví dụ: trong một sản phẩm phần mềm bệnh viện, có một ca sử dụng với diễn viên Y tá và một ca sử dụng khác với diễn viên Y tế nhân viên không phải là ý kiến hay, vì tất cả y tá đều là nhân viên y tế, nhưng một số nhân viên y tế (chẳng hạn như bác sĩ) không phải là y tá. . Sẽ tốt hơn nếu có các diễn viên Bác sĩ và Y tá. Ngoài ra, nhân viên y tế có thể được định nghĩa với hai chuyên môn, Bác sĩ và Y tá. Điều này được mô tả trong Hình 11.2. Trong Phần 7.7, người ta đã chỉ ra rằng thừa kế là một trường hợp đặc biệt của khái quát hóa. Tổng quát hóa đã được áp dụng cho các lớp trong Phần 7.7. Hình 11.2 cho thấy cách tổng quát hóa cũng có thể được áp dụng cho các tác nhân.)

* 1. Initial Requirements (Yêu cầu ban đầu)

To determine the client’s requirements, initial requirements are drawn up based on the initial business model. Then, as the understanding of the domain and the business model is refined on the basis of further discussions with the client, the requirements are refined.

The requirements are dynamic. That is, there are frequent changes not just to the require- ments themselves but also to the attitudes of the development team, client, and future users toward each requirement. For example, a particular requirement may first appear to the development team to be optional. After further analysis, that requirement may now seem to be critically important. However, after discussion with the client, the requirement is rejected. A good way to handle these frequent changes is to maintain a list of likely require- ments, together with use cases of the requirements that have been agreed to by the members of the development team and approved by the client.

It is important to bear in mind that the object-oriented paradigm is iterative and the glossary, the business model, or the requirements therefore may have to be modified at any time. In particular, additions to the requirements list, modifications to items already on the list, and removal of items from the list can be triggered by a wide variety of events, ranging from a casual remark made by a user to a suggestion from the client at a formal meeting of the systems analysts on the requirements team. Any such change may trigger correspond- ing changes to the business model.

Requirements fall into two categories, functional and nonfunctional. A **functional requirement** specifies an action that the target product must be able to perform. Func- tional requirements are often expressed in terms of inputs and outputs: Given a spe- cific input, the functional requirement stipulates what the output must be. Conversely, a **nonfunctional requirement** (or **quality requirement**) specifies properties of the tar- get product itself, such as **platform constraints** (“The software product shall run under Linux”), **response times** (“On average, queries of Type 3B shall be answered within 2.5 seconds”), or **reliability** (“The software product shall run 99.5 percent of the time”).

Functional requirements are handled while the requirements and analysis workflows are being performed, whereas some nonfunctional requirements may have to wait until the design workflow. The reason is that, to be able to handle certain nonfunctional requirements, detailed knowledge about the target software product may be needed, and this knowledge is usually not available until the requirements and analysis workflows have been completed (see Problems 11.1 and 11.2). However, wherever possible, nonfunctional requirements should also be handled during the requirements and analysis workflows.

The requirements workflow is now illustrated by a running case study.

(Dịch: Để xác định yêu cầu của khách hàng, các yêu cầu ban đầu được đưa ra dựa trên mô hình kinh doanh ban đầu. Sau đó, khi hiểu biết về miền và mô hình kinh doanh được tinh chỉnh trên cơ sở thảo luận thêm với khách hàng, các yêu cầu sẽ được tinh chỉnh.

Các yêu cầu là động. Có nghĩa là, có những thay đổi thường xuyên không chỉ đối với bản thân các yêu cầu mà còn về thái độ của nhóm phát triển, khách hàng và người dùng tương lai đối với từng yêu cầu. Ví dụ: một yêu cầu cụ thể trước tiên có thể xuất hiện đối với nhóm phát triển là tùy chọn. Sau khi phân tích sâu hơn, yêu cầu đó bây giờ có vẻ là cực kỳ quan trọng. Tuy nhiên, sau khi thảo luận với khách hàng, yêu cầu bị từ chối. Một cách tốt để xử lý những thay đổi thường xuyên này là duy trì một danh sách các yêu cầu có thể xảy ra, cùng với các trường hợp sử dụng của các yêu cầu đã được các thành viên của nhóm phát triển đồng ý và được khách hàng chấp thuận.

Điều quan trọng cần lưu ý là mô hình hướng đối tượng là lặp đi lặp lại và bảng thuật ngữ, mô hình kinh doanh hoặc các yêu cầu do đó có thể phải được sửa đổi bất cứ lúc nào. Cụ thể, việc bổ sung vào danh sách yêu cầu, sửa đổi các mục đã có trong danh sách và xóa các mục khỏi danh sách có thể được kích hoạt bởi nhiều sự kiện khác nhau, từ nhận xét thông thường của người dùng đến đề xuất từ ​​khách hàng tại một cuộc họp chính thức của các nhà phân tích hệ thống trong nhóm yêu cầu. Bất kỳ thay đổi nào như vậy có thể kích hoạt các thay đổi tương ứng với mô hình kinh doanh.

Các yêu cầu được chia thành hai loại, chức năng và không chức năng. Một yêu cầu chức năng chỉ định một hành động mà sản phẩm mục tiêu phải có thể thực hiện. Yêu cầu chức năng thường được thể hiện dưới dạng đầu vào và đầu ra: Với một đầu vào cụ thể, yêu cầu chức năng quy định đầu ra phải là gì. Ngược lại, một yêu cầu phi chức năng (hoặc yêu cầu chất lượng) chỉ định các thuộc tính của chính sản phẩm tar- get, chẳng hạn như các ràng buộc nền tảng (“Sản phẩm phần mềm sẽ chạy trong Linux”), thời gian phản hồi (“Trung bình, các truy vấn thuộc Loại 3B sẽ được trả lời trong vòng 2,5 giây ”), hoặc độ tin cậy (“ Sản phẩm phần mềm sẽ chạy 99,5 phần trăm thời gian ”).

Các yêu cầu chức năng được xử lý trong khi các yêu cầu và quy trình phân tích đang được thực hiện, trong khi một số yêu cầu phi chức năng có thể phải đợi cho đến khi quy trình thiết kế. Lý do là để có thể xử lý các yêu cầu phi chức năng nhất định, có thể cần kiến ​​thức chi tiết về sản phẩm phần mềm mục tiêu và kiến ​​thức này thường không có sẵn cho đến khi các yêu cầu và quy trình phân tích đã được hoàn thành (xem Vấn đề 11.1 và 11.2). Tuy nhiên, bất cứ khi nào có thể, các yêu cầu phi chức năng cũng cần được xử lý trong quá trình yêu cầu và quy trình phân tích.

Quy trình công việc yêu cầu hiện được minh họa bằng một nghiên cứu điển hình đang chạy.)

*Case Study*



* 1. Initial Understanding of the Domain: The MSG Foundation Case Study

When Martha Stockton Greengage died at the age of 87, she left her entire $2.3 billion fortune to charity. Specifically, her will set up the Martha Stockton Greengage (MSG) Foundation to assist young couples in purchasing their own homes by providing low- cost loans.

To reduce operating expenses, the trustees of the MSG Foundation are investigat- ing computerization. Because none of the trustees has any experience with comput- ers, they decide to commission a small software development organization to imple- ment a pilot project, namely, a software product that will perform the calculations needed to determine how much money is available each week to purchase homes.

The first step, as always, is to understand the application domain, home mortgages in this instance. Not many people can afford to pay cash to buy a home. Instead, they pay a small percentage of the purchase price out of their own savings and borrow the rest of the money. This type of loan, where real estate is pledged as security for the loan, is termed a **mortgage** (see Just in Case You Wanted to Know Box 11.2).

For example, suppose that someone wishes to buy a house for $100,000. (Many houses nowadays cost much more than that, particularly in the larger cities, but the round number makes the arithmetic easier.) The person buying the house pays a **deposit** of (say) 10 percent, or $10,000, and borrows the remaining $90,000 from a financial insti- tution such as a bank or a savings and loan company in the form of a mortgage for that amount. Accordingly, the **principal** (or **capital**) borrowed is $90,000.

Suppose that the terms of the mortgage are that the loan is to be repaid in monthly installments over 30 years at an interest rate of 7.5 percent per annum (or 0.625 percent



**Just in Case You Wanted to Know**

**Box 11.2**

Have you ever wondered why the word *mortgage* is pronounced “more gidge” with the accent on the first syl- lable? The word, which was first used in Middle English in the fourteenth century, comes from the Old French word *mort* meaning “dead” and the Germanic word *gage* meaning “a pledge,” that is, a promise to forfeit property if the debt is not paid. Strangely enough, a mortgage is a “dead pledge” in two different senses. If the loan is not repaid, the property is forfeited, or “dead” to the borrower, forever. And if the loan is repaid, then the promise to repay is dead. This two-way explanation was first given by the English judge Sir Edward Coke (1552–1634).

And the strange pronunciation? The final letter in a French word like *mort* is silent—hence the “more.” And the suffix *-age* is frequently pronounced “idge” in English. Examples include the words *carriage, marriage, disparage,* and *encourage*.

per month). Each month, the borrower pays the finance company $629.30. Part of this amount is the interest on the outstanding balance; the rest is used to reduce the principal. This monthly payment is therefore often referred to as **P & I** (**principal and interest**). For example, in the first month the outstanding balance is $90,000. Monthly interest at 0.625 percent on $90,000 is $562.50. The remainder of the P & I payment of $629.30, namely $66.80, is used to reduce the principal. Consequently, at the end of the first month, after the first payment has been made, only $89,933.20 is owed to the finance company.

The interest for the second month is 0.625 percent of $89,933.20, or $562.08. The P & I payment is $629.30, as before, and the balance of the P & I payment (now

$67.22) again is used to reduce the principal, this time to $89,865.98.

After 15 years (180 months), the monthly P & I payment is still $629.30, but now the principal has been reduced to $67,881.61. The monthly interest on $67,881.61 is

$424.26, so the remaining $205.04 of the P & I payment is used to reduce the princi- pal. After 30 years (360 months), the entire loan will have been repaid.

The finance company wants to be certain that it will be repaid the $90,000 it is owed, plus interest. It ensures this in a number of different ways.

* First, the borrower signs a legal document (the mortgage deed) that states that, if the monthly payments are not made, the finance company may sell the house and use the proceeds to pay off the outstanding balance of the loan.
* Second, the finance company requires the borrower to insure the house, so that if (say) the house burns down, the insurance company will cover the loss and the check from the insurance company will then be used to repay the loan. The insurance premium is usually paid once a year by the finance company. To obtain the money for the premium from the borrower, the finance company requires the borrower to pay monthly insurance installments. It deposits the installments in an **escrow account**, essentially a savings account managed by the finance com- pany. When the annual insurance premium is due, the money is taken from the escrow account. Real-estate taxes paid on a home are treated the same way; that is, monthly installments are deposited in the escrow account and the annual real- estate tax payment is made from that account.
* Third, the finance company wants to be sure that the borrower can afford to pay for the mortgage. Typically, a mortgage will not be granted if the total monthly

**FIGURE 11.3** The initial glossary of the MSG Foundation case study.

**Balance:** the amount of the loan still owing

**Capital:** synonym for principal

**Closing costs:** other costs involved in buying a house, such as legal costs and various taxes

**Deposit:** an initial installment toward the total cost of the house

**Escrow account:** a savings account managed by the finance company into which the weekly installments toward the annual insurance premium and annual real-estate tax payment are deposited, and from which the annual insurance premium and the annual real-estate tax payment are paid

**Interest:** a cost of borrowing money, computed as a fraction of the amount owing

**Mortgage:** a loan in which real estate is pledged as security for the loan

**P & I:** abbreviation for “principal and interest“

**Points:** a cost of borrowing money, computed as a fraction of the total amount borrowed

**Principal:** the lump sum borrowed

**Principal and interest:** an installment payment consisting of the interest plus the fraction of the principal for that installment

payment (P & I plus insurance plus real-estate taxes) exceeds 28 percent of the borrower’s total income.

In addition to the monthly payments, the finance company almost always wants to be paid a lump sum up front in return for lending the money to the borrower. Typi- cally, the finance company will want 2 percent of the principal (“2 **points**”). In the case of the $90,000 loan, this amounts to $1800.

Finally, there are other costs involved in buying a house, such as legal costs and various taxes. Consequently, when the contract to buy the $100,000 house is signed (when the deal is “closed”), the **closing costs** (legal costs, taxes, and so on) plus the points can easily amount to $7000.

The initial glossary of the MSG Foundation domain is shown in Figure 11.3. The initial business model of the MSG Foundation case study is now constructed.

###### Case Study



* 1. Initial Business Model: The MSG Foundation Case Study

Members of the development organization interview various managers and staff members of the MSG Foundation and discover the way the Foundation operates. At the start of each week, the MSG Foundation estimates how much money will be

**FIGURE 11.4** The Estimate Funds Available for Week use case of the initial business model of the MSG Foundation case study.

**MSG Foundation Information System**

Estimate Funds Available for Week

**MSG Staff Member**

available that week to fund mortgages. Couples whose income is too low to afford a standard mortgage to buy a home can apply at any time to the MSG Foundation for a mortgage. An MSG Foundation staff member first determines whether the couple qualifies for an MSG mortgage and then determines whether the MSG Foundation still has sufficient funds on hand that week to purchase the home. If so, the mortgage is granted and the weekly mortgage repayment is computed according to the MSG Foundation’s rules. This repayment amount may vary from week to week, depending on the couple’s current income.

The corresponding part of the business model consists of three use cases: Esti- mate Funds Available for Week, Apply for an MSG Mortgage, and Compute Weekly Repayment Amount. These use cases are shown in Figures 11.4, 11.5, and 11.6, respectively, and the corresponding initial **use-case descriptions** appear in Figures 11.7, 11.8, and 11.9, respectively.

Consider the use case Apply for an MSG Mortgage (Figure 11.5). The actor on the right is **Applicants**. But is **Applicants** really an actor? Recall from Section

11.4.3 that an actor is a user of a software product. However, applicants do not use the software product. They fill in a form. Their answers are then entered into the software product by an MSG staff member. In addition, they may ask questions of the staff mem- ber or answer questions put to them by the staff member. But regardless of their interac- tions with MSG staff members, applicants never interact with the software product.1

However,

* First, the **Applicants** initiate the use case. That is, if a couple does not apply for a mortgage, this use case never occurs.
* Second, the information that the **MSG Staff Member** gives to the software product is provided by the **Applicants**.
* Third, in a sense, the real actor is the **Applicants**; the **MSG Staff Member** is merely an agent of the **Applicants**.

For all these reasons, **Applicants** is indeed an actor.

Now consider Figure 11.6, which depicts the use case Compute Weekly Repayment Amount. The actor on the right is now **Borrowers**. Once an

1 This will change if the MSG Foundation ever decides to accept applications over the Web. Specifically, **Applicants** will then become the only actor in Figure 10.6; **MSG Staff Member** will no longer play a role.

**FIGURE 11.5** The Apply for an MSG Mortgage use case of the initial business model of the MSG Foundation case study.

**MSG Foundation Information System**

Apply for an MSG Mortgage

**MSG Staff Member**

**Applicants**

**FIGURE 11.6** The Compute Weekly Repayment Amount use case of the initial business model of the MSG Foundation case study.

**MSG Foundation Information System**

Compute Weekly Repayment Amount

**MSG Staff Member**

**Borrowers**



**FIGURE 11.7** The description of the Estimate Funds Available for Week use case of the initial business model of the MSG Foundation case study.

**Step-by-Step Description**

Not applicable at this initial stage.

**Brief Description**

The Estimate Funds Available for Week use case enables an MSG Foundation staff member to estimate how much money the Foundation has available that week to fund mortgages.

application has been granted, the couple who applied for the mortgage (the **Applicants**) become **Borrowers**. But even as borrowers they do not interact with the software product. As before, only MSG staff members can enter informa- tion into the software product. Nevertheless, again the use case is initiated by actor **Borrowers** and again the information entered by the **MSG Staff Member** is supplied by the **Borrowers**. Accordingly, **Borrowers** is indeed an actor in the use case shown in Figure 11.6.

Another aspect of the MSG Foundation business model concerns the investments of the MSG Foundation. At this initial stage details are not yet known regarding the

**FIGURE 11.8** The description of the Apply for an MSG Mortgage use case of the initial business model of the MSG Foundation case study.

**Step-by-Step Description**

Not applicable at this initial stage.

**Brief Description**

When a couple applies for a mortgage, the Apply for an MSG Mortgage use case enables an MSG Foundation staff member to determine whether they qualify for an MSG mortgage and, if so, whether funds are currently available for the mortgage.

**FIGURE 11.9** The description of the Compute Weekly Repayment Amount use case of the initial business model of the MSG Foundation case study.

**Step-by-Step Description**

Not applicable at this initial stage.

**Brief Description**

The Compute Weekly Repayment Amount use case enables an MSG Foundation staff member to compute how much borrowers have to repay each week.

**FIGURE 11.10** The Manage an Investment use case of the initial business model of the MSG Foundation case study.

**MSG Foundation Information System**

Manage an Investment

**MSG Staff Member**

buying and selling of investments or how investment income becomes available for mortgages, but it is certainly clear that the use case Manage an Investment shown in Figure 11.10 is an essential part of the initial business model. The initial description appears in Figure 11.11; in a future iteration, details of how investments are handled will be inserted.

For conciseness, the four use cases of Figures 11.4, 11.5, 11.6, and 11.10 are com- bined into the **use-case diagram** of Figure 11.12.

Now the initial requirements have to be drawn up.

**FIGURE 11.11** The description of the Manage an Investment

use case of the initial business model of the MSG Foundation case study.

**Step-by-Step Description**

Not applicable at this initial stage.

**Brief Description**

The Manage an Investment use case enables an MSG Foundation staff member to buy and sell investments and manage the investment portfolio.

**FIGURE 11.12** The use-case diagram of the initial business model of the MSG Foundation case study.

**MSG Foundation Information System**

Estimate Funds Available for Week

**MSG Staff Member**

###### Case Study



Apply for an MSG Mortgage

Compute Weekly Repayment Amount

Manage an Investment

**Applicants**

**Borrowers**

* 1. Initial Requirements: The MSG Foundation Case Study

The four use cases of Figure 11.12 comprise the business model of the MSG Foundation. However, it is not immediately obvious whether they are all require- ments of the MSG Foundation software product that is to be developed. Recall that what the client *wants* is “a pilot project, namely, a software product that will perform

the calculations needed to determine how much money is available each week to purchase homes.” As always, the task of the developers is to determine, with the aid of the client, what the client *needs*. At this early stage, however, there is not enough information at the analysts’ disposal to be able to decide whether just this “pilot proj- ect” will be what is needed. In situations like this, the best way to proceed is to draw up the initial requirements on the basis of what the client wants, and then iterate.

Accordingly, each of the use cases of Figure 11.12 in turn is considered. Use case Estimate Funds Available for Week is obviously part of the initial requirements. On the other hand, Apply for an MSG Mortgage does not seem to have anything to do with the pilot project, so it is excluded from the initial requirements. At first sight, the third use case, Compute Weekly Repayment Amount, seems equally irrelevant to the pilot project. However, the pilot project deals with the “money that is available each week to purchase homes.” Part of that money surely comes from the weekly repayment of existing mortgages, so the third use case is indeed part of the initial requirements. The fourth use case, Manage an Investment, is also part of the initial requirements for a similar reason—income from investments also must be used to fund new mortgages.

The initial requirements then consist of three use cases and their descriptions, namely, Estimate Funds Available for Week (Figures 11.4 and 11.7), Compute Weekly Repayment Amount (Figures 11.6 and 11.9), and Manage an Investment (Figures 11.10 and 11.11). These three use cases appear in Figure 11.13.

The next step is to iterate the requirements workflow; that is, the steps are per- formed again to obtain a better model of the client’s needs.

**FIGURE 11.13** The use-case diagram of the initial requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Estimate Funds Available for Week



**MSG Staff Member**

Compute Weekly Repayment Amount

Manage an Investment

**Borrowers**

###### Case Study



* 1. Continuing the Requirements Workflow: The MSG Foundation Case Study

Armed with domain knowledge and familiarity with the initial business model, mem- bers of the development team now interview the MSG Foundation managers and staff in greater depth. They discover the following information.

The MSG Foundation grants a 100 percent mortgage to buy a home under the fol- lowing conditions:

* The couple has been married for at least 1 year but not more than 10 years.
* Both husband and wife are gainfully employed. Specifically, proof must be provided that both were employed full time for at least 48 weeks of the preceding year.
* The price of the home must be below the published median price for homes in that area for the past 12 months.
* The installments on a fixed-rate, 30-year, 90 percent mortgage would exceed 28 percent of their combined gross income and/or they do not have sufficient savings to pay 10 percent of the cost of the home plus $7000. (The $7000 is an estimate of the additional costs involved, including closing costs and points.)
* The Foundation has sufficient funds to purchase the home; this is described later in more detail.

If the application is approved, then the amount that the couple should pay the MSG Foundation every week for the next 30 years is the total of the principal and interest payment, which never changes over the life of the mortgage, and the escrow payment, which is —1 nd of the sum of the annual real-estate tax and the annual homeowner’s insurance premium. If this total is greater than 28 percent of the couple’s gross weekly income, then the MSG Foundation will pay the difference in the form of a grant. Consequently, the mortgage is paid in full each week, but the couple will never have to pay more than 28 percent of their combined gross income.

52

The couple must provide a copy of their income tax return each year so that the MSG Foundation has proof of their previous year’s income. In addition, the couple may file copies of pay slips as proof of current gross income. The amount the couple has to pay for their mortgage may therefore vary from week to week.

The MSG Foundation uses the following algorithm to determine whether it has the funds to approve a mortgage application:

1. At the beginning of each week, the estimated annual income from its investments is computed and divided by 52.
2. The estimated annual MSG Foundation operating expenses are divided by 52.
3. The total of the estimated mortgage payments for that week is computed.
4. The total of the estimated grants for that week is computed.
5. The amount available at the beginning of the week is then (Item 1) — (Item 2) +

(Item 3) — (Item 4).

1. During the week, if the cost of the home is no more than the amount available for mortgages, then the MSG Foundation deems that it has the funds needed to pur- chase the home; the amount available for mortgages that week is reduced by the cost of that home.
2. At the end of each week, the MSG Foundation investment advisors invest any unspent funds.

To keep the cost of the pilot project as low as possible, the developers are told that only those data items needed for the weekly funds computation should be incorporated into the software product. The rest can be added later if the MSG Foundation decides to computerize all aspects of its operation. Therefore, only three types of data are needed, namely, investment data, operating expenses data, and mortgage data.

With regard to investments, the following data are required:

Item number. Item name.

Estimated annual return. (This figure is updated whenever new information becomes available. On average, this occurs about four times a year.)

Date estimated annual return was last updated.

With regard to operating expenses, the following data are required:

Estimated annual operating expenses. (This figure is currently determined four times a year.)

Date estimated annual operating expenses were last updated. For each mortgage, the following data are required:

Account number.

Last name of mortgagees. Original purchase price of home. Date mortgage was issued.

Weekly principal and interest payment. Current combined gross weekly income.

Date combined gross weekly income was last updated. Annual real-estate tax.

Date annual real-estate tax was last updated. Annual homeowner’s insurance premium.

Date annual homeowner’s insurance premium was last updated.

In the course of further discussions with MSG managers, the developers learn that three types of reports are needed:

The results of the funds computation for the week.

A listing of all investments (to be printed on request). A listing of all mortgages (to be printed on request).

###### Case Study

* 1. Revising the Requirements: The MSG Foundation Case Study

Recall that the initial requirements model (Section 11.8) includes three use cases, namely, Estimate Funds Available for Week, Compute Weekly Repayment Amount, and Manage an Investment. These use cases are shown in Figure 11.13. Now, in the light of the additional information that has been received, the initial requirements can be revised.

The formula given in Section 11.9 for determining how much money is available at the beginning of a week is as follows:

1. The estimated annual income from investments is computed and divided by 52.
2. The estimated annual MSG Foundation operating expenses are divided by 52.
3. The total of the estimated mortgage payments for that week is computed.
4. The total of the estimated grants for that week is computed.
5. The amount available is then (Item 1) — (Item 2) + (Item 3) — (Item 4). Consider each of these items in turn.
6. *Estimated annual income from investments*. For each investment in turn, sum the estimated annual return on each investment, and divide the result by 52. To do this, an additional use case is needed, namely, Estimate Investment Income for Week. (Use case Manage an Investment is still needed for adding, deleting, and modifying investments.) This new use case is depicted in Figure 11.14 and described in Figure 11.15. In Figure 11.14, the dashed line with the open arrowhead labeled «include» denotes that use case Estimate Investment Income for Week is part of use case Estimate Funds Available for Week. The resulting first iteration of the revised use-case diagram is shown in Figure 11.16 with the new use case shaded.
7. *Estimated annual operating expenses*. Up to now, the estimated annual operating expenses have not been considered. To incorporate these expenses, two additional

**FIGURE 11.14** The Estimate Investment Income for Week use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**



**MSG Staff Member**

Estimate Funds Available for Week

«include»

Estimate Investment Income for Week

**FIGURE 11.15** The description of the Estimate Investment Income for Week use case of the revised requirements of the MSG Foundation case study.

|  |
| --- |
| **Brief Description**  The Estimate Investment Income for Week use case enables the Estimate Funds Available for Week use case to estimate how much investment income is available for this week. |
| **Step-by-Step Description**   1. For each investment, extract the estimated annual return on that investment. 2. Sum the values extracted in Step 1 and divide the result by 52. |

**FIGURE 11.16** The first iteration of the use-case diagram of the revised requirements of the MSG Foundation case study. The new use case is shaded.

**MSG Foundation Information System**

**MSG Staff Member**

Estimate Funds Available for Week

Compute Weekly Repayment Amount

Manage an Investment

«include»

Estimate Investment Income for Week

**Borrowers**



use cases are needed. Use case Update Estimated Annual Operating Expenses models adjustments to the value of the estimated annual operating expenses, and use case Estimate Operating Expenses for Week provides the estimate of the operating expenses that is required. The use cases are shown in Figures 11.17 through 11.20. In Figure 11.19, use case Estimate Operating Expenses for Week is similarly part of use case Estimate Funds Available for Week, as indicated by the dashed line with the open arrowhead labeled «include». The resulting second iteration of the revised use-case diagram is shown in Figure 11.21. The two new use cases, Estimate Operating Expenses for Week and Update Estimated Annual Operating Expenses, are shaded.

1. *Total estimated mortgage payments for the week*. (See item 4.)

**FIGURE 11.17** The Update Estimated Annual Operating Expenses use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Update Estimated

Annual Operating Expenses

**MSG Staff Member**

**FIGURE 11.18** The description of the Update Estimated Annual Operating Expenses use case of the revised requirements of the MSG Foundation case study.

**Step-by-Step Description**

1. Update the estimated annual operating expenses.

**Brief Description**

The Update Estimated Annual Operating Expenses use case enables an MSG Foundation staff member to update the estimated annual operating expenses.

**FIGURE 11.19** The Estimate Operating Expenses for Week use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Estimate Funds Available for Week

«include»

Estimate Operating Expenses for Week

**MSG Staff Member**

1. *Total estimated grant payments for the week*. The weekly repayment amount from use case Compute Weekly Repayment Amount is the total estimated mortgage payment less the estimated total grant payment. In other words, use case Compute Weekly Repayment Amount models the computation of both the estimated mortgage payment and the estimated grant payment for each mort- gage separately. Summing these separate quantities will yield the total estimated mortgage payments for the week as well as the total estimated grant payments for

**FIGURE 11.20** The description of the Estimate Operating Expenses for Week use case of the revised requirements of the MSG Foundation case study.

**Brief Description**

The Estimate Operating Expenses for Week use case enables the Estimate Funds Available for Week use case to estimate the operating expenses for the week.

**Step-by-Step Description**

1. Divide the estimated annual operating expenses by 52.

**FIGURE 11.21** The second iteration of the use-case diagram of the revised requirements of the MSG Foundation case study. The two new use cases, Estimate Operating Expenses for Week and Update Estimated Annual Operating Expenses, are shaded.

**MSG Foundation Information System**

**MSG Staff Member**

Estimate Funds Available for Week

«include»

Compute Weekly Repayment Amount

«include»

Manage an Investment

Update Estimated

Annual Operating Expenses

Estimate Investment Income for Week

Estimate Operating Expenses for Week

**Borrowers**



the week. However, Compute Weekly Repayment Amount also models the borrowers changing the amount of their weekly income. Accordingly, Compute Weekly Repayment Amount needs to be split into two separate use cases, namely, Estimate Payments and Grants for Week and Update Borrowers’ Weekly Income. The two new use cases are described in

**FIGURE 11.22** The Estimate Payments and Grants for Week use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Estimate Funds Available for Week

«include»

Estimate Payments and Grants for Week

**MSG Staff Member**

Figures 11.22 through 11.25. Once more, one of the new use cases, namely, Estimate Payments and Grants for Week, is part of use case Esti- mate Funds Available for Week, as indicated by the dashed line with the open arrowhead labeled «include» in Figure 11.22. The resulting third iteration of the revised use-case diagram is shown in Figure 11.26 with the two use cases derived from use case Compute Weekly Repayment Amount shaded.

Consider Figure 11.26 again. Use case Estimate Funds Available for Week models the computation that uses the data obtained from three other use cases,

**FIGURE 11.23** The description of the Estimate Payments and Grants for Week use case of the revised requirements of the MSG Foundation case study.

52

and the annual homeowner’s insurance premium.

* 1. Compute 28 percent of the couple’s current gross weekly income.
  2. If the result of Step 1.1 is greater than the result of Step 1.2, then the mortgage payment for this week is the result of Step 1.2, and the amount of the grant for this week is the difference between the result of Step 1.1 and the result of Step 1.2.
  3. Otherwise, the mortgage payment for this week is the result of Step 1.1 and there is no grant this week.

1. Summing the mortgage payments of Steps 1.3 and 1.4 yields the estimated mortgage payments for the week.
2. Summing the grant payments of Step 1.3 yields the estimated grant payments for the week.

**Step-by-Step Description**

1. For each mortgage:
   1. The amount to be paid this week is the total of the principal and interest payment and —1 nd of the sum of the annual real-estate tax

**Brief Description**

The Estimate Payments and Grants for Week use case enables the Estimate Funds Available for Week use case to estimate the total estimated mortgage payments paid by borrowers to the MSG Foundation for this week and the total estimated grants paid by the MSG Foundation for this week.

**FIGURE 11.24** The Update Borrowers’ Weekly Income use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Update Borrowers’ Weekly Income

**MSG Staff Member**

**Borrowers**



**FIGURE 11.25** The description of the Update Borrowers’ Weekly Income use case of the revised requirements of the MSG Foundation case study.

**Step-by-Step Description**

1. Update the borrower’s weekly income.

**Brief Description**

The Update Borrowers’ Weekly Income use case enables an MSG Foundation staff member to update the weekly income of a couple who have borrowed money from the Foundation.

namely, Estimate Investment Income for Week, Estimate Oper- ating Expenses for Week, and Estimate Payments and Grants for Week. This is shown in Figure 11.27, which shows the second iteration of the use case Estimate Funds Available for Week; this figure has been ex- tracted from the use-case diagram of Figure 11.26. Figure 11.28 is the corresponding description of the use case.

Why is it so important to indicate the «**include**» **relationship** in UML diagrams? For example, Figure 11.29 shows two versions of Figure 11.22, the correct version on top and an incorrect version below. The top diagram correctly models use case Estimate Funds Available for Week as part of use case Estimate Payments and Grants for Week. The bottom diagram of Figure 11.29 models use cases Estimate Funds Available for Week and Estimate Payments and Grants for Week as two independent use cases. However, as stated in Section 11.4.3, a use case models an interaction between the software product itself and users of the software product (actors). This is fine for use case Estimate Funds Available for Week. However, use case Estimate Payments and Grants for Week does not interact with an actor and, therefore, cannot be a use case in its own right. Instead, it is a portion of use case Estimate Funds Available for Week, as reflected in the top diagram of Figure 11.29.

**FIGURE 11.26** The third iteration of the use-case diagram of the revised requirements of the MSG Foundation case study. The two use cases derived from use case Compute Weekly Repayment Amount are shaded.

«include»



**MSG Staff Member**

«include»

**rrowers**

**FIGURE 11.27** The second iteration of the Estimate Funds Available for Week use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Estimate Investment Income for Week

Estimate Estimate Funds «include» Operating Available for Expenses for

Week Week

Update Borrowers’ Weekly Income

Estimate Payments and Grants for Week

**Bo**

Manage an Investment

Update Estimated

Annual Operating Expenses

**MSG Foundation Information System**

Estimate Investment Income for Week

Estimate Funds Available for Week

«include»

Estimate Operating Expenses for Week

Estimate Payments and Grants for Week

«include»

«include»

**MSG Staff Member**

**FIGURE 11.28** The second iteration of the description of the Estimate Funds Available for Week use case of the revised requirements of the MSG Foundation case study.

|  |
| --- |
| **Brief Description**  The Estimate Funds Available for Week use case enables an MSG Foundation staff member to estimate how much money the Foundation has available that week to fund mortgages. |
| **Step-by-Step Description**   1. Determine the estimated income from investments for the week utilizing use case   Estimate Investment Income for Week.   1. Determine the operating expenses for the week utilizing use case Estimate Operating Expenses for Week. 2. Determine the total estimated mortgage payments for the week utilizing use case   Estimate Payments and Grants for Week.   1. Determine the total estimated grants for the week utilizing use case Estimate Payments and Grants for Week. 2. Add the results of Steps 1 and 3 and subtract the results of Steps 2 and 4. This is the   total amount available for mortgages for the current week. |

**FIGURE 11.29** Correct (top) and incorrect (bottom) versions of Figure 11.22.

**MSG Foundation Information System**

Estimate Funds Available for Week

«include»

Estimate Payments and Grants for Week

**MSG Staff Member**



**MSG Staff Member**

**MSG Foundation Information System**

Estimate Funds Available for

Week

Estimate Payments and Grants for Week

###### Case Study



* 1. The Test Workflow: The MSG Foundation Case Study

A common side effect of the iterative-and-incremental life-cycle model is that details that have been correctly postponed somehow get forgotten. That is one of the many reasons why continual testing is essential. In this instance, the details of the use case Manage an Investment have been overlooked. This is remedied in Figures

11.30 and 11.31.

Further review brings to light the omission of use case Manage a Mortgage to model the addition of a new mortgage, the modification of an existing mort- gage, or the removal of an existing mortgage, analogous to use case Manage an Investment. Figures 11.32 and 11.33 correct this omission, and the fourth itera- tion of the revised use-case diagram is shown in Figure 11.34 with the new use case, Manage a Mortgage, shaded.

**FIGURE 11.30** The Manage an Investment use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Manage an Investment

**MSG Staff Member**

**FIGURE 11.31** The description of the Manage an Investment use case of the revised requirements of the MSG Foundation case study.

**Step-by-Step Description**

1. Add, modify, or delete an investment.

**Brief Description**

The Manage an Investment use case enables an MSG Foundation staff member to add and delete investments and manage the investment portfolio.

**FIGURE 11.32** The Manage a Mortgage use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Manage a Mortgage

**MSG Staff Member**

**FIGURE 11.33** The description of the Manage a Mortgage use case of the revised requirements of the MSG Foundation case study.

**Step-by-Step Description**

1. Add, modify, or delete a mortgage.

**Brief Description**

The Manage a Mortgage use case enables an MSG Foundation staff member to add and delete mortgages and manage the mortgage portfolio.

Furthermore, the use case for printing the various reports has also been over- looked. Accordingly, use case Produce a Report, which models the printing of the three reports, is added. The details of the use case appear in Figures 11.35 and

11.36. The fifth iteration of the revised use-case diagram is shown in Figure 11.37 with the new use case, Produce a Report, shaded.

The revised requirements are checked yet again, and two new problems are uncov-

ered. First, a use case has been partially duplicated. Second, two of the use cases need to be reorganized.

The first change to be made is to remove the partially duplicated use case. Con- sider the use case Manage a Mortgage (Figures 11.32 and 11.33). As stated in Figure 11.33, one of the actions of this use case is to modify a mortgage. Now con- sider the use case Update Borrowers’ Weekly Income (Figures 11.24 and 11.25). The only purpose of this use case (Figure 11.25) is to update the borrowers’ weekly income. But the borrowers’ weekly income is an attribute of the mortgage. That is, use case Manage a Mortgage already includes the use case Update Borrowers’ Weekly Income. Accordingly, use case Update Borrowers’ Weekly Income is superfluous and should be deleted. The result is shown in Figure 11.38, the sixth iteration of the revised use-case diagram. The modified use case, Manage a Mortgage, is shaded.

**FIGURE 11.34** The fourth iteration of the use-case diagram of the revised requirements of the MSG Foundation case study. The new use case, Manage a Mortgage, is shaded.

**MSG Foundation Information System**

Estimate Investment Income for Week

Estimate Funds Available for Week

«include»

Estimate Operating Expenses for Week

Update Borrowers' Weekly Income

Estimate Payments and Grants for Week

**Bo**

Manage an Investment

Manage a Mortgage

Update Estimated

Annual Operating Expenses

«include»

**rrowers**

**MSG Staff Member**

«include»

**FIGURE 11.35** The Produce a Report use case of the revised requirements of the MSG Foundation case study.

**MSG Foundation Information System**

Produce a Report

**MSG Staff Member**

**FIGURE 11.36** The description of the Produce a Report use case of the revised requirements of the MSG Foundation case study.

**Brief Description**

The Produce a Report use case enables an MSG Foundation staff member to print the results of the weekly computation of funds

available for new mortgages or to print a listing of all investments or all mortgages.

**Step-by-Step Description**

1. The following reports must be generated:
   1. Investments report—printed on demand:

The information system prints a list of all investments. For each investment, the following attributes are printed:

Item number Item name

Estimated annual return

Date estimated annual return was last updated

* 1. Mortgages report—printed on demand:

The information system prints a list of all mortgages. For each mortgage, the following attributes are printed:

Account number Name of mortgagee Original price of home

Date mortgage was issued Principal and interest payment

Current combined gross weekly income

Date current combined gross weekly income was last updated

Annual real-estate tax

Date annual real-estate tax was last updated Annual homeowner’s insurance premium

Date annual homeowner’s insurance premium was last updated

* 1. Results of the weekly computation—printed each week:

The information system prints the total amount available for new mortgages during the current week

This is the first iteration that has resulted in a decrement rather than an increment. That is, this is the first time in this book that the result of an iteration has been to delete an artifact (the Update Borrowers’ Weekly Income use case). In fact, deletion occurs all too often, namely, whenever a mistake is made. Sometimes an incorrect artifact can be fixed, but frequently an artifact has to be deleted. The key point is that, when a fault is discovered, there is no need to abandon everything done to date and start the whole requirements process from scratch. Instead, an attempt is

**FIGURE 11.37** The fifth iteration of the use-case diagram of the revised requirements of the MSG Foundation case study. The new use case, Produce a Report, is shaded.

**MSG Foundation Information System**

Estimate Investment Income for Week

Estimate Estimate Funds «include» Operating Available for Expenses for

Week Week

Update Borrowers’ Weekly Income

Estimate Payments and Grants for Week

**Bo**

Manage an Investment

Manage a Mortgage

Update Estimated Annual

Operating Expenses

Produce a Report

«include»

**rrowers**

**MSG Staff Member**

«include»

made to fix the current iteration, as was done in this case study. If this strategy fails (because the mistake really is serious), we backtrack to the previous iteration and try to find a better way to go forward from there.

The second change that must be made to improve the requirements is to reorga- nize two use cases. Consider the descriptions of the use cases Estimate Funds Available for Week (Figure 11.28) and Produce a Report (Figure 11.36). Suppose that an MSG staff member wants to determine the funds available for the current week. Use case Estimate Funds Available for Week performs

**FIGURE 11.38** The sixth iteration of the use-case diagram of the revised requirements of the MSG Foundation case study. The modified use case, Manage a Mortgage, is shaded.

**MSG Foundation Information System**

Estimate Investment Income for Week

«include»

**MSG Staff Member**

Estimate Funds Available for Week

Manage an Investment

«include»

Manage a Mortgage

Update Estimated

Annual Operating Expenses

Produce a Report

«include»

Estimate Operating Expenses for Week

Estimate Payments and Grants for Week

**Borrowers**



the calculation, and Step 1.3 of use case Produce a Report prints out the re- sult of the computation. This is ridiculous. After all, there is no point in estimating the funds available unless the results are printed out.

In other words, Step 1.3 of Produce a Report needs to be moved from the description of that use case to the end of the description of use case Estimate Funds Available for Week. This does not change the use cases themselves (Figures 11.27 and 11.35) or the current use-case diagram (Figure 11.38), but the descriptions of the two use cases (Figures 11.28 and 11.36) have to be modified. The resulting modified descriptions are shown in Figures 11.39 and 11.40.

**FIGURE 11.39** The second iteration of the description of the Produce a Report use case of the revised requirements of the MSG Foundation case study.

|  |
| --- |
| **Brief Description**  The Produce a Report use case enables an MSG Foundation staff member to print a listing of all investments or all mortgages. |
| **Step-by-Step Description**   1. The following reports must be generated:    1. Investments report—printed on demand:   The information system prints a list of all investments. For each investment, the following attributes are printed:  Item number Item name  Estimated annual return  Date estimated annual return was last updated   * 1. Mortgages report—printed on demand:   The information system prints a list of all mortgages. For each mortgage, the following attributes are printed:  Account number Name of mortgagee Original price of home  Date mortgage was issued Principal and interest payment  Current combined gross weekly income  Date current combined gross weekly income was last updated Annual real-estate tax  Date annual real-estate tax was last updated Annual homeowner’s insurance premium  Date annual homeowner’s insurance premium was last updated |

**FIGURE 11.40** The third iteration of the description of the Estimate Funds Available for Week use case of the revised requirements of the MSG Foundation case study.

|  |
| --- |
| **Brief Description**  The Estimate Funds Available for Week use case enables an MSG Foundation staff member to estimate how much money the Foundation has available that week to fund mortgages. |
| **Step-by-Step Description**   1. Determine the estimated income from investments for the week utilizing use case   Estimate Investment Income for Week.   1. Determine the operating expenses for the week utilizing use case   Estimate Operating Expenses for Week.   1. Determine the total estimated mortgage payments for the week utilizing use case   Estimate Payments and Grants for Week.   1. Determine the total estimated grants for the week utilizing use case Estimate Payments and Grants for Week. 2. Add the results of Steps 1 and 3 and subtract the results of Steps 2 and 4. This is the   total amount available for mortgages for the current week.   1. Print the total amount available for new mortgages during the current week. |

**344**

**FIGURE 11.41** Use case Print Tax Form is part of three other use cases.

**MSG Foundation Information System**

Prepare Form 1040

«include»

**Tax Preparer**

«include»

Prepare Form 1040A

Prepare Form 1040EZ

«include»

Print Tax Form



Now the use-case diagram can be improved still further. Consider the top four use cases in Figure 11.38. The three use cases on the right, namely, Estimate Investment Income for Week, Estimate Operating Expenses for Week, and Estimate Payments and Grants for Week, are part of the use case Estimate Funds Available for Week. The usual reason for an «include» relationship is when one use case is part of two or more other use cases. For example, Figure 11.41 shows that use case Print Tax Form is part of use cases Prepare Form 1040, Prepare Form 1040A, and Prepare Form 1040EZ, the three primary U.S. tax forms for individuals. In this situation, it makes sense to retain Print Tax Form as an independent use case. Incorporat- ing the operations of Print Tax Form into the other three use cases would mean triplicating that use case.

With regard to Figure 11.38, however, all the included use cases are part of only one use case, namely, Estimate Funds Available for Week—there is no duplication. Accordingly, it makes sense to incorporate those three «include» use cases into Estimate Funds Available for Week, as shown in Figure 11.42, the seventh iteration of the use-case diagram. The resulting fourth iteration of the description of the Estimate Funds Available for Week use case is shown in Figure 11.43.

Now the requirements appear to be correct.

* First, they correspond to what the client has requested.
* Second, there do not seem to be any faults.
* Third, at this stage it would seem that what the client wants coincides with what the client needs.

**FIGURE 11.42** The seventh iteration of the use-case diagram of the revised requirements of the MSG Foundation case study. The modified use case, Estimate Funds Available for Week, is shaded.

**MSG Foundation Information System**

Estimate Funds Available for Week

Manage an Investment

Manage a Mortgage

**MSG Staff Member**

Update Estimated

Annual Operating Expenses

Produce a Report

**Borrowers**



Accordingly, the requirements workflow appears to be complete, for now. Nev- ertheless, it is certainly possible that, during subsequent workflows, additional re- quirements may surface. Also, it may be necessary to split one or more of the five use cases into additional use cases. For example, in a future iteration the Produce a Report use case described in Figure 11.36 may be split into two separate use cases, one for the investments report, the other for the mortgages report. But for now, everything seems to be satisfactory.

This concludes the description of the requirements workflow for the MSG Foun- dation case study.

**FIGURE 11.43** The fourth iteration of the description of the use case Estimate Funds Available for Week of the revised requirements of the MSG Foundation case study.

**Brief Description**

The Estimate Funds Available for Week use case enables an MSG Foundation staff member to estimate how much money the Foundation has available that week to fund mortgages.

**Step-by-Step Description**

1. For each investment, extract the estimated annual return on that investment. Summing the separate returns and dividing the result by 52 yields the estimated investment income for the week.
2. Determine the estimated MSG Foundation operating expenses for the week by extracting the estimated annual MSG Foundation operating expenses and dividing by 52.
3. For each mortgage:
   1. The amount to be paid this week is the total of the principal and interest payment and —1 nd of the sum of the annual real-estate tax and the annual homeowner’s insurance premium.

52

* 1. Compute 28 percent of the couple’s current gross weekly income.
  2. If the result of Step 3.1 is greater than the result of Step 3.2, then the mortgage payment for this week is the result of Step 3.2, and the amount of the grant for this week is the difference between the result of Step 3.1 and the result of Step 3.2.
  3. Otherwise, the mortgage payment for this week is the result of Step 3.1, and there is no grant this week.

1. Summing the mortgage payments of Steps 3.3 and 3.4 yields the estimated total mortgage payments for the week.
2. Summing the grant payments of Step 3.3 yields the estimated total grant payments for the week.
3. Add the results of Steps 1 and 4 and subtract the results of Steps 2 and 5. This is the total amount available for mortgages for the current week.
4. Print the total amount available for new mortgages during the current week.
   1. **The Classical Requirements Phase**

On the one hand, there is no such thing as “object-oriented requirements,” nor should there be such a thing. The aim of the requirements workflow is to determine the client’s needs, that is, what the functionality of the target system should be. The requirements workflow has nothing to do with how the product is to be built. From this viewpoint, it makes no sense to refer to the classical paradigm or the object-oriented paradigm within the context

of the requirements workflow, any more than one can refer to a classical or object-oriented user manual. After all, the user manual describes the steps to be followed by the user when running the software product and has nothing to do with how the product was built. In the same way, the requirements workflow results in a statement of what the product is to do; the way that the product will be built does not enter into it.

On the other hand, the entire approach of Sections 11.2 through 11.11 is object oriented in nature in that it is model oriented. The use cases, together with their descriptions, form the basis of the requirements workflow. As is shown throughout Part B of this book, model- ing is the essence of the object-oriented paradigm.

However, modeling in general (and UML modeling in particular) is not part of the clas- sical paradigm. The classical requirements phase starts with requirements elicitation fol- lowed by requirements analysis, similarly to the object-oriented paradigm (Sections 11.3 through 11.4.2). But from that point on, the two paradigms diverge. Instead of building models, the next step in the classical requirements phase is to draw up a list of require- ments. The usual step after that is to build a rapid prototype that implements the key func- tionality underlying those requirements; this is described in Section 11.13. The client and future users of the target software product then experiment with the rapid prototype until the requirements team members are satisfied that the rapid prototype exhibits the key func- tionality of the software product the client needs.

Building a rapid prototype for the product as a whole is not part of the object-oriented paradigm, for the reasons given in Section 13.18. However, it is strongly advisable to build a rapid prototype of the user interface, as will be described.

* 1. Rapid Prototyping

A **rapid prototype** is hastily built software that exhibits the key functionality of the target product. For example, a product that helps to manage an apartment complex must incorporate an input screen that allows the user to enter details of a new tenant and print an occupancy report for each month. These aspects are incorporated into the rapid prototype. However, error-checking capabilities, file-updating routines, and complex tax computa- tions probably are not included. The key point is that a rapid prototype reflects the func- tionality the client sees, such as input screens and reports, but omits “hidden” aspects such as file updating. (For a different way of looking at rapid prototypes, see Just in Case You Wanted to Know Box 11.3.)

The client and intended users of the product now experiment with the rapid prototype, while members of the development team watch and take notes. Based on their hands-on experience, users tell the developers how the rapid prototype satisfies their needs and, more important, identify the areas that need improvement. The developers change the rapid pro- totype until both sides are convinced that the needs of the client are accurately encapsulated in the rapid prototype. The rapid prototype is then used as the basis for drawing up the specifications.

An important aspect of the rapid prototyping model is embodied in the word *rapid*. The whole idea is to build the rapid prototype as quickly as possible. After all, the purpose of the rapid prototype is to provide the client an understanding of the product, and the sooner the better. It does not matter if the rapid prototype hardly works, if it crashes every few



**Just in Case You Wanted to Know Box 11.3**

The idea of constructing models to show key aspects of a product goes back a long time. For example, a 1618 painting by Domenico Cresti (known as “Il Passignano” because he was born in the town of Passignano in the Chianti region of Italy) shows Michelangelo presenting a wooden model of his design for St. Peter’s (in Rome) to Pope Paul IV. Such architectural models could be huge; a model of an earlier design proposal for St. Peter’s by the architect Bramante is more than 20 feet long on each side.

Architectural models were used for a number of different purposes. First, as depicted in the Cresti painting (now hanging in Casa Buonarroti in Florence), models were used to try to interest a client in funding a project. This is analogous to the use of a rapid prototype to determine the client’s real needs. Second, in an age before architectural drawings, the model showed the builder the structure of the building and indicated to the stonemasons how the building was to be decorated. This is similar to the way we now build a rapid pro- totype of the user interface, as described in Section 11.13.

It is not a good idea, however, to draw too close a parallel between such architec- tural models and software rapid prototypes. Rapid prototypes are used during the classical requirements phase to elicit the client’s needs. Unlike architectural models, they are not used to represent either the architectural design or the detailed design; the design is pro- duced two phases later, that is, during the classical design phase.

minutes, or if the screen layouts are less than perfect. The purpose of the rapid prototype is to enable the client and the developers to agree as quickly as possible on what the product is to do. Therefore, any imperfections in the rapid prototype may be ignored, provided that they do not seriously impair the functionality of the rapid prototype and thereby give a misleading impression of how the product behaves.

A second major aspect of the rapid prototyping model is that the rapid prototype must be built for change. If the first version of the rapid prototype is not what the client needs, then the prototype must be transformed rapidly into a second version that, it is hoped, better satisfies the client’s requirements. To achieve rapid development throughout the rapid prototyping process, fourth-generation languages (4GL) and interpreted languages, such as Smalltalk, Prolog, and Lisp, have been used for rapid prototyping purposes. Pop- ular rapid prototyping languages of today include HTML and Perl. Concerns have been expressed about the maintainability of certain interpreted languages, but from the view- point of rapid prototyping this is irrelevant. All that counts is this: Can a given language be used to produce a rapid prototype? And, can the rapid prototype be changed quickly? If the answer to both questions is Yes, then that language is probably a good candidate for rapid prototyping.

Rapid prototyping is particularly effective when developing the user interface to a prod- uct. This use is discussed in Section 11.14.

* 1. Human Factors

It is important that both the client and the future users of the product interact with the rapid prototype of the user interface. Encouraging users to experiment with the human–computer interface (HCI) greatly reduces the risk that the finished product will have to be altered.

In particular, this experimentation helps achieve user-friendliness, a vital objective for all software products.

The term **user-friendliness** refers to the ease with which human beings can commu- nicate with the software product. If users have difficulty in learning how to use a product or find the screens confusing or irritating, then they will either not use the product or use it incorrectly. To try to eliminate this problem, menu-driven products were introduced. Instead of having to enter a command such as Perform computation or Print service rate report, the user merely has to select from a set of possible responses, such as

1. Perform computation
2. Print service rate report
3. Select view to be graphed

In this example, the user enters 1, 2, or 3 to invoke the corresponding command. Nowadays, instead of simply displaying lines of text, HCIs employ graphics. Windows,

icons, and pull-down menus are components of a **graphical user interface (GUI)** (see Just in Case You Wanted to Know Box 11.4). Because of the plethora of windowing systems, stan- dards such as X Window have evolved. Also, **point-and-click** selection is now the norm. The user moves a mouse (that is, a handheld pointing device) to move the screen cursor to the desired response (“point”), and pushes a mouse button (“click”) to select that response.

However, even when the target product employs modern technology, the designers must never forget that the product is to be used by human beings. In other words, the HCI design- ers must consider **human factors** such as size of letters, capitalization, color, line length, and the number of lines on the screen.

Another example of human factors applies to the preceding menu. If the user chooses option 3. Select view to be graphed, then another menu appears with another list of choices. Unless a menu-driven system is thoughtfully designed, there is the danger that users will encounter a lengthy sequence of menus to achieve even a relatively simple operation. This delay can anger users, sometimes causing them to make inappropriate menu selections. Also, the HCI must allow the user to change a previous selection without having to return to the top-level menu and start again. This problem can exist even when a GUI is used because many graphical user interfaces are essentially a series of menus displayed in an attractive screen format.

Sometimes it is impossible for a single user interface to cater to all users. For example, if a product is to be used by both computer professionals and high-school dropouts with no previous computer experience, then it is preferable that two different sets of HCIs be designed, each carefully tailored to the skill level and psychological profile of its intended users. This technique can be extended by incorporating sets of user interfaces requiring var- ied levels of sophistication. If the product deduces that the user would be more comfortable with a less sophisticated user interface, perhaps because the user is making frequent mis- takes or is continually invoking help facilities, then the user is automatically shown screens that are more appropriate to his or her current skill level. But, as the user becomes more familiar with the product, streamlined screens that provide less information are displayed, leading to speedier completion. This automated approach reduces user frustration and leads to increased productivity [Schach and Wood, 1986].

Many benefits can accrue when human factors are taken into account during the design of an HCI, including reduced learning times and lower error rates. Although help facilities



##### Just in Case You Wanted to Know Box 11.4

The GUI was invented at Xerox’s Palo Alto Research Centre (PARC) in the 1970s. At that time it was called the WIMP interface, where WIMP stands for either Window, Icon, Mouse, and Pull-down menu, or Window, Icon, Menu, and Pointing device, depending on whom you believe. The first commercial computer with a WIMP interface was the Xerox 8010 (“Star”), launched in 1981.

The GUI achieved popularity with the release of the Apple Lisa (1983) and the Apple Macintosh (1984). The Macintosh design team had been invited by PARC researchers to see their WIMP interface, and several PARC employees subsequently left PARC and went to work at Apple on the GUIs for the Lisa and the Macintosh. The Apple software engineers considerably extended and improved the WIMP interface.

Microsoft soon implemented a GUI of its own. But in 1988, Apple sued Microsoft for copyright infringement of the Lisa and Macintosh GUIs, claiming that the copyright of the “look and feel” of its GUIs had been violated. The court case lasted 4 years before almost all of Apple’s claims were denied, primarily due to a license Apple had negotiated with Microsoft for Windows 1.0. Ironically, midway through the case, Xerox filed its own lawsuit against Apple, claiming Apple had infringed the copyrights Xerox held on its GUIs. The Xerox case was dismissed because the three-year statute of limitations had passed. Related legal disputes between Apple and Microsoft continued until 1997. At that time, all remain- ing copyright infringement issues were settled by negotiation. Microsoft invested $150 million in nonvoting Apple stock, and the two companies signed a patent cross-licensing agreement.

The GUI became the de facto user interface in 1995 with the introduction of Microsoft Windows 95.

must always be provided, they are utilized less with a carefully designed HCI. This, too, increases productivity. Uniformity of HCI appearance across a product or group of products can result in users intuitively knowing how to use a screen that they have never seen before because it is similar to other screens with which they are familiar. Designers of Macintosh software have taken this principle into account; this is one of the many reasons that software for the Macintosh is generally so user-friendly.

It has been suggested that simple common sense is all that is needed to design a user- friendly HCI. Whether or not this charge is true, it is essential that a rapid prototype of the HCI of every product be constructed. Intended users of the product can experiment with the rapid prototype of the HCI and inform the designers whether the target product indeed is user-friendly, that is, whether the designers have taken the necessary human factors into account.

In Section 11.15, reuse is discussed within the context of rapid prototyping.

* 1. Reusing the Rapid Prototype

After the rapid prototype has been built, it is discarded early in the software process. An alternate, but generally unwise, way of proceeding is to develop and refine the rapid pro- totype until it becomes the product. In theory, this approach should lead to fast software development; after all, instead of throwing away the code constituting the rapid prototype,

along with the knowledge built into it, the rapid prototype is converted into the final prod- uct. The first problem with this form of the rapid prototyping model follows from the fact that, in the course of refining the rapid prototype, changes have to be made to a working product. This is an expensive way to proceed, as shown in Figure 1.6. A second problem is that a primary objective when constructing a rapid prototype is speed of building. A rapid prototype is (correctly) hurriedly put together, rather than carefully specified, designed, and implemented. In the absence of specification and design documents, the resulting code is difficult and expensive to maintain. It might seem wasteful to construct a rapid prototype and then throw it away and design the product from scratch, but it is far cheaper in both the short term and the long term to do this rather than try to convert a rapid prototype into production quality software [Brooks, 1975].

Another reason for discarding the rapid prototype is the issue of performance, particu- larly of real-time systems. To ensure that time constraints are met, it is necessary to design the product carefully. In contrast, a rapid prototype is constructed to display key functional- ity to the client; performance issues are not handled. As a result, if an attempt is made to refine a rapid prototype into a delivered product, it is unlikely that response times and other timing constraints will be met.

One way of ensuring that the rapid prototype is thrown away and the product is properly designed and implemented is to build the rapid prototype in a different language from that of the product. For example, the client may specify that the product must be implemented in Java. If the rapid prototype is implemented in HTML, for example, it must be discarded. First, the rapid prototype is implemented in HTML and refined until the client is satisfied that it does everything, or almost everything, the target product is to do. Next, the product is designed, relying on the knowledge and skills acquired in constructing the rapid prototype. Finally, the design is implemented in Java and the tested product handed over to the client in the usual way.

Nevertheless, there is one instance when it is permissible to refine a rapid prototype or, more specifically, portions of the rapid prototype. When portions of the rapid prototype are computer generated, those portions may be used in the final product. For example, user interfaces are often a key aspect of a rapid prototype. When CASE tools such as screen generators and report generators (Section 5.7 and summarized in Section 10.8) have been utilized to generate the user interfaces, those portions of the rapid prototype may indeed be used as part of production-quality software.

The desire not to “waste” the rapid prototype has resulted in a modified version of the rapid prototyping model being adopted by some organizations. Here, management decides *before* the rapid prototype is built that portions may be utilized in the final prod- uct, provided those portions pass the same quality assurance tests as other software com- ponents. Therefore, after the rapid prototype is complete, those sections the developers wish to continue to use must pass design and code inspections. This approach goes beyond rapid prototyping. For example, components that are of sufficiently high quality to pass design and code inspections are not usually found in a rapid prototype. Furthermore, design documents are not part of classic rapid prototyping. Nevertheless, this hybrid approach is attractive to some organizations hoping to recover some of the time and money invested in the rapid prototype. However, to ensure that the quality of the code is sufficiently high, the rapid prototype has to be built somewhat more slowly than is cus- tomary for a “rapid” prototype.

* 1. CASE Tools for the Requirements Workflow

The many UML diagrams in this chapter reflect the importance of having a graphical tool to assist with the requirements workflow. That is, what is needed is a drawing tool that enables the user to draw the relevant UML diagrams with ease. Such a tool has two major strengths.

* First, while iterating it is generally far easier to change a diagram stored in such a tool than to redraw the diagram by hand.
* Second, when a CASE tool of this kind is used, the details of the product are stored in the CASE tool itself. Therefore, the documentation is always available and up to date.

One weakness of such CASE tools is that they are not always user-friendly. A powerful graphical workbench or environment has so much functionality that it generally has a steep learning curve, and even experienced users sometimes have difficulty remembering how to achieve a particular outcome. A second weakness is that it is almost impossible to program a computer to draw UML diagrams that are as aesthetically pleasing as diagrams drawn by hand by humans. One alternative is to spend a considerable amount of time “tweaking” a dia- gram created by a tool. However, this approach is sometimes as slow as drawing the diagrams by hand. Worse, the constraints of many graphical CASE tools are such that, no matter how much time and effort is put into a diagram, it can never look as polished as a hand-drawn diagram. A third problem is that many CASE tools are expensive. It is not unusual to have to pay $5000 or more per user for a comprehensive CASE tool. On the other hand, a number of open-source CASE tools of this type can be downloaded at no cost. Overall, the two bulleted strengths of CASE tools listed in this section outweigh these weaknesses.

Many of the classical graphical CASE workbenches and environments, such as System Architect and Software through Pictures, have been extended to support UML diagrams. In addition, there are object-oriented CASE workbenches and environments, such as IBM Rational Rose and Together. There are also open-source CASE tools of this type, including ArgoUML.

* 1. Metrics for the Requirements Workflow

A key feature of the requirements workflow is how rapidly the requirements team deter- mines the client’s real needs. So, a useful metric during this workflow is a measure of requirements volatility. Keeping a record of how frequently the requirements change during the requirements workflow gives management a way of determining the rate at which the requirements team converges on the actual requirements of the product. This metric has the further advantage that it can be applied to any requirements elicitation technique, such as interviewing or forms analysis.

Another measure of how well the requirements team is doing its job is the number of requirements that change during the rest of the software development process. For each such change in requirements, it should be recorded whether that change was initiated by the client or the developers. If a large number of changes in requirements are initi- ated by the developers during the analysis, design, and subsequent workflows, then it is clear that the process used by the team to carry out the requirements workflow should

be thoroughly reviewed. Conversely, if the client makes repeated changes to the require- ments during subsequent workflows, then this metric can be used to warn the client that the moving-target problem can adversely affect the project, and future changes should be held to a minimum.

* 1. Challenges of the Requirements Workflow

Like every other workflow of the software development process, potential problems and pitfalls are associated with the requirements workflow. First, it is essential to have the wholehearted cooperation of the potential users of the target product from the beginning of the process. Individuals often feel threatened by computerization, fearing that the computer will take their jobs. There is some truth to that fear. Over the past 30 years or so, the impact of computerization has been to reduce the need for unskilled workers but also to generate jobs for skilled workers. Overall, the number of well-paying employment opportunities created as a direct consequence of computerization has far exceeded the number of rela- tively unskilled jobs made redundant, as evidenced by both decreased unemployment rates and increased average compensation. But the unparalleled economic growth of so many countries worldwide as a direct or indirect consequence of the so-called Computer Age in no way can compensate for the negative impact on those individuals who lose their jobs as a result of computerization.

It is essential that every member of the requirements team be aware at all times that the members of the client organization with whom they interact in all probability are deeply concerned about the potential impact of the target software product on their jobs. In the worst case, employees may deliberately give misleading or wrong information to try to en- sure that the product does not meet the client’s needs and, hence, protect those employees’ jobs. But, even with no sabotage of this kind, some members of the client organization may be less than helpful simply because they have a vague feeling of being threatened by computerization.

Another challenge of the requirements workflow is the ability to **negotiate**. For exam- ple, it is often essential to scale down what the client wants. Not surprisingly, almost every client would love to have a software product that can do everything that might conceivably be needed. Such a product would take an unacceptably long time to build and cost far more than the client considers reasonable. Therefore, it often is necessary to persuade the client to accept less (sometimes far less) than he or she wants. Computing the costs and benefits (see Section 5.2 and summarized in Section 10.6) of each requirement in dispute can help in this regard.

Another example of the negotiating skill needed is the ability to arrive at a compromise among managers regarding the functionality of the target product. For example, a cunning manager may attempt to extend his or her power by including a requirement that can be implemented only by incorporating into his or her areas of responsibility certain business functions currently the responsibility of another manager. Not surprisingly, the other man- ager will object strongly on discovering what is going on. The requirements team must sit down with both managers and resolve the issue.

A third challenge of the requirements workflow is that, in many organizations, the individuals who possess information the requirements team needs to elicit, simply lack the



* **Iterate**

Obtain an understanding of the domain. Draw up the business model.

Draw up the requirements.

* **Until** the requirements are satisfactory.

**Box 11.1**

**How to Perform the Requirements Workflow**

time to meet for in-depth discussions. When this happens, the team must inform the client, who then must decide which is more important, the individuals’ current job responsibilities or the software product to be constructed. And, if the client fails to insist that the software product comes first, the developers may have no alternative but to withdraw from a project all but doomed to failure.

Finally, flexibility and objectivity are essential for requirements elicitation. It is vital that the members of the requirements team approach each interview with no preconceived ideas. In particular, an interviewer must never make assumptions about the requirements as a result of earlier interviews, and then conduct subsequent interviews in the light of those assumptions. Instead, an interviewer must consciously suppress any information gleaned at previous interviews and conduct each interview in an impartial way. Making premature assumptions regarding the requirements is dangerous; making any assumptions during the requirements workflow regarding the software product to be built can be disastrous.

The chapter concludes with How to Perform Box 11.1, which summarizes the steps of the requirements workflow.

**Chapter Review**

The chapter begins with a description of the importance of determining the client’s needs (Section 11.1), followed by an overview of the requirements workflow (Section 11.2). In Section 11.3, the need to understand the domain is described. How to draw up the business model is described in Sec- tion 11.4. Interviewing and other techniques of requirements extraction are discussed in Sections

11.4.1 and 11.4.2. The business model is modeled using use cases, which are introduced in 11.4.3. Drawing up the initial requirements is described in Section 11.5. The requirements workflow of the MSG Foundation case study is presented in the next six sections. Obtaining an initial understanding of the domain is described in Section 11.6; the initial business model and the initial requirements are presented in Sections 11.7 and 11.8, respectively. The requirements are then refined in Sections

11.9 and 11.10. Finally, the test workflow for the MSG Foundation case study is described (Section 11.11). In Section 11.12, the classical requirements phase is contrasted with the requirements work- flow of the Unified Process. Rapid prototyping is then discussed in greater detail in Sections 11.13 and 11.14; in the latter section, the importance of constructing a rapid prototype for the user interface is stressed. In Section 11.15, a warning is given not to reuse a rapid prototype. CASE tools for the requirements workflow (Section 11.16) and metrics for the requirements workflow (Section 11.17) are then discussed. The chapter concludes with a description of challenges of the requirements phase (Section 11.18).

An overview of the MSG Foundation case study in this chapter appears in Figure 11.44.

**FIGURE 11.44** Overview of the MSG Foundation case study for Chapter 11.

|  |
| --- |
| Initial understanding of the domain Section 11.6 Initial glossary Figure 11.3  lnitial business model Section 11.7  lnitial use-case diagram Figure 11.12  Initial requirements Sections 11.8, 11.9  Revised requirements Section 11.10  Second iteration of the use-case diagram Figure 11.21  Third iteration of the use-case diagram Figure 11.26 Test workflow Section 11.11  Fourth iteration of the use-case diagram Figure 11.34  Fifth iteration of the use-case diagram Figure 11.37  Sixth iteration of the use-case diagram Figure 11.38  Seventh iteration of the use-case diagram Figure 11.42 |

**For Further Reading**

[Jackson, 1995] is an excellent introduction to requirements analysis. [Thayer and Dorfman, 1999] is a collection of papers on requirements analysis. Berry [2004] suggests that the ripple effect of the inevitable changes to the requirements is the reason why there cannot be a software engineering silver bullet (Just in Case You Wanted to Know Box 3.4). The use of cost–benefit analysis in setting priorities among requirements is described in [Karlsson and Ryan, 1997]. Nonfunctional requirements are discussed in [Cysneiros and do Prado Leite, 2004] and [Grego- riades and Sutcliffe, 2005].

The requirements workflow of the Unified Process is described in detail in Chapters 6 and 7 of [Jacobson, Booch, and Rumbaugh, 1999]. Misuse cases (use cases that model interactions that the software should prevent) are described in [I. Alexander, 2003].

The importance of prototyping is described in [Schrage, 2004].

Having an effective requirements process has a positive effect on the entire life cycle. This is demonstrated in [Damian and Chisan, 2006] by means of a case study of a large-scale soft- ware project. An analysis of agile approaches to requirements engineering appears in [Cao and Ramesh, 2008].

A variety of articles on requirements appear in the May–June 2006 issue of *IEEE Software*; [Ebert, 2006] is of particular interest. Further articles appear in the March–April 2007 issue. The March–April 2008 issue of *IEEE Software* contains articles on nonfunctional requirements (“quality requirements”), including [Blaine and Cleland-Huang, 2008], [Glinz, 2008], and [Feather et al., 2008].

The annual Requirements Engineering conference is an excellent source of information.

A classic work on user interface design is [Shneiderman, 2003]. Methods for achieving good user interfaces are described in [Holzinger, 2005]. Articles on user interfaces can be found in the June 2008 issue of *Communications of the ACM*. The proceedings of the Annual Conference on Human Factors in Computer Systems (sponsored by ACM SIGCHI) are a valuable source of information on wide-ranging aspects of human factors.

Chapter 11 *Requirements* **357**

**Key Terms** actor *318*

application domain *314*

business model *316*

direct observation *317*

domain *314*

form *317*

functional requirement *320*

glossary *315*

graphical user interface (GUI) *350*

human factors *350*

«include» relationship *335*

model *318*

negotiation *354*

nonfunctional requirement *320*

platform constraint *320* point and click *350* quality requirement *320*

questionnaire *317*

rapid prototype *348*

reliability *320*

requirements analysis *315*

requirements capture *315*

requirements elicitation *315*

requirements engineering *315*

requirements workflow *314*

response time *320*

structured interview *316*

unstructured interview *316*

use case *318*

use-case description *323*

use-case diagram *325*

user-friendliness *350*

videotape camera *317*

**Case Study Key Terms**

capital *320*

closing costs *322*

deposit *320*

escrow account *321*

interest *321*

mortgage *320*

P & I *321*

points *322*

principal *320*

**Problems** 11.1 Give a nonfunctional requirement that can be handled without having detailed knowledge about the target software product.

* 1. Now, give a nonfunctional requirement that can be handled only after the requirements work- flow has been completed.
  2. Your client has stipulated that open-source software is to be used. Is this a functional or non- functional requirement? How early in the life-cycle model can this requirement be handled? Explain your answer.
  3. Your client has stipulated that all documentation has to be written in both English and isiNde- bele. Is this a functional or nonfunctional requirement? How early in the life-cycle model can this requirement be handled? Explain your answer.
  4. Distinguish between a *use case* and a *use-case diagram*.
  5. You have been asked to develop a logistics automation system for a ship chandler. How would you perform the domain analysis?
  6. What do you consider to be the most important questions when interviewing the ship chandler of Problem 11.6?
  7. Distinguish between a *user* and an *actor*.
  8. When performing the requirements workflow for a bank payroll product, why is it inadvisable to model the product with **Tellers** and **Employees** as actors?
  9. Draw a flowchart representing the requirements workflow.
  10. Why does the same couple appear as two different actors (**Applicants** and **Borrowers**) in the use-case diagram of Figure 11.12?
  11. Noting that only MSG Foundation staff members can use the software product, why do

**Applicants** and **Borrowers** appear as actors in the use-case diagram of Figure 11.12?

* 1. Use a spreadsheet to show that, at the end of 30 years, monthly installments of $629.30 will pay off a loan for $90,000 with interest compounded monthly at an annual rate of 7.5 percent.
  2. Explain why annual real-estate taxes and insurance premiums are generally paid from an escrow account, rather than directly by the borrower (mortgagee).

**358** Part B *The Workflows of the Software Life Cycle*

* 1. Suppose that the MSG Foundation decides that it wants its software product to include the mortgage application process. Give the description of the Apply for an MSG Mort- gage use case. Give as many details as you can.
  2. Sections 11.9 and 11.10 describe the restructuring of the use cases of the MSG Foundation. How would this restructuring change if, as in Problem 11.15, the Apply for an MSG Mortgage use case had been included in the requirements model?
  3. You have just joined Langfoss & Yosemite Software as a software manager. Langfoss & Yosemite has been developing accounting software for small businesses for many years using the waterfall model, usually with some success. On the basis of your experience, you think that the Unified Process is a far superior way of developing software. Write a report addressed to the vice-president for software development explaining why you believe the organization should switch to the Unified Process. Remember that vice-presidents do not like reports that are more than half a page in length.
  4. You are the vice-president for software development of Langfoss & Yosemite. Reply to the report of Problem 11.17.
  5. What is the result if a rapid prototype is not constructed rapidly?
  6. Why is there an advantage to using an interpreted language for implementing a rapid proto- type, rather than a compiled language? Is there a disadvantage?
  7. (Analysis and Design Project) Perform the requirements workflow for the automated library circulation system of Problem 8.7.
  8. (Analysis and Design Project) Perform the requirements workflow for the product for deter- mining whether a bank statement is correct of Problem 8.8.
  9. (Analysis and Design Project) Perform the requirements workflow for the automated teller machine (ATM) of Problem 8.9.
  10. (Term Project) Perform the requirements workflow for the Chocoholics Anonymous project in Appendix A.
  11. (Case Study) The trustees of the MSG Foundation have decided to expand their activities by providing scholarships for higher education to children of current borrowers with a suffi- ciently high grade-point average. Draw the use case Apply for an MSG Scholarship. Give the description of the use case, providing as much detail as you can.
  12. (Case Study) A report of all scholarships awarded during the past year (Problem 11.25) has to be generated. Modify Figures 11.35 and 11.36 appropriately to incorporate this additional report.
  13. (Case Study) Using the information in Sections 11.6 through 11.11, construct a rapid prototype for the MSG Foundation case study. Use the software and hardware specified by your instructor.
  14. (Readings in Software Engineering) Your instructor will distribute copies of [Damian and Chisan, 2006]. In what ways did reading this article change your views on the importance of the requirements workflow?

**References** [I. Alexander, 2003] I. ALEXANDER, “Misuse Cases: Use Cases with Hostile Intent,” *IEEE Software*

**20** (January–February 2003), pp. 58–66.

[Berry, 2004] D. M. BERRY, “The Inevitable Pain of Software Development: Why There Is No Sil- ver Bullet,” in: *Radical Innovations of Software and Systems Engineering in the Future*, Lecture Notes in Computer Science, Vol. 2941, Springer-Verlag, Berlin, 2004, pp. 50–74.

[Blaine and Cleland-Huang, 2008] J. D. BLAINE AND J. CLELAND-HUANG, “Software Quality Requirements: How to Balance Competing Priorities,” *IEEE Software* **25** (March–April 2008), pp. 22–24.

Chapter 11 *Requirements* **359**

[Brooks, 1975] F. P. BROOKS, JR., *The Mythical Man-Month: Essays on Software Engineering,* Addison-Wesley, Reading, MA, 1975; Twentieth Anniversary Edition, Addison-Wesley, Reading, MA, 1995.

[Cao and Ramesh, 2008] L. CAO AND B. RAMESH, “Agile Requirements Engineering Practices: An Empirical Study,” *IEEE Software* **25** (January–February 2008), pp. 60–67.

[Cysneiros and do Prado Leite, 2004] L. M. CYSNEIROS AND J. C. S. DO PRADO LEITE, “Nonfunctional Requirements: From Elicitation to Conceptual Models,” *IEEE Transactions on Software Engi- neering* **30** (May 2004), pp. 328–50.

[Damian and Chisan, 2006] D. DAMIAN AND J. CHISAN, “An Empirical Study of the Complex Rela- tionships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management,” *IEEE Transactions on Software Engineering* **32** (July 2006), pp. 433–53.

[Ebert, 2006] C. EBERT, “Understanding the Product Life Cycle: Four Key Requirements Engineer- ing Techniques,” *IEEE Software* **23** (May–June 2006), pp. 19–25.

[Feather et al., 2008] M. S. FEATHER, S. L. CORNFORD, K. A. HICKS, J. D. KIPER, AND T. MENZIES,

“A Broad, Quantitative Model for Making Early Requirements Decisions,” *IEEE Software* **25**

(March–April 2008), pp. 49–56.

[Glinz, 2008] M. GLINZ, “A Risk-Based, Value-Oriented Approach to Quality Requirements,” *IEEE Software* **25** (March–April 2008), pp. 34–41.

[Gregoriades and Sutcliffe, 2005] A. GREGORIADES AND A. SUTCLIFFE, “Scenario-Based Assessment of Nonfunctional Requirements,” *IEEE Transactions on Software Engineering* **31** (May 2005), pp. 392–409.

[Holzinger, 2005] A. HOLZINGER, “Usability Engineering Methods for Software Developers,” *Com- munications of the ACM* **48** (January 2005), pp. 71–74.

[Jackson, 1995] M. JACKSON, *Software Requirements and Specifications: A Lexicon of Practice, Prin- ciples and Prejudices,* Ad